

Transferarbeit

Softwareentwicklungsprojekt

Studiengruppe Wirtschaftsinformatik

Lerngruppe:

Brzank, Patrick
Buchner, Maximilian
Steinbauer, Isabel

Thema 1: Bergrettung

Erstgutachter: Note: ...

Zweitgutachter: Note:.....

Transferarbeit Note:

Eigenständigkeitserklärung

Wir haben die vorliegende Transferarbeit im Rahmen der Ausbildungssequenz Vorgehensmodelle und Softwareentwicklungsprojekt selbstständig verfasst und keine anderen als die angegebenen Quellen, Tools und Hilfsmittel benutzt. Die vorliegende Arbeit oder Teile daraus sind noch nicht Leistungsnachweis einer vorangegangenen Prüfung gewesen bzw. sind entsprechend als Quelle gekennzeichnet.

München, den 27.02.2020

Maximilian Buchner

Maximilian Buchner

München, den 27.02.2020

Patrick Brzank

Patrick Brzank

München, den 27.02.2020

Isabel Steinbauer

Isabel Steinbauer

Inhaltsverzeichnis

	Seite
Eigenständigkeitserklärung.....	II
Inhaltsverzeichnis.....	III
Abbildungsverzeichnis.....	V
Abkürzungen.....	VI
1 Pflichtenheft.....	1
1.1 Zielbestimmung.....	1
1.1.1 Muss-Kriterien.....	1
1.1.2 Kann-Kriterien.....	1
1.1.3 Abgrenzungskriterien.....	1
1.2 Einsatz.....	1
1.2.1 Anwendungsbereiche.....	1
1.2.2 Zielgruppen.....	1
1.2.3 Betriebsbedingungen.....	1
1.3 Umgebung.....	2
1.3.1 Software.....	2
1.3.2 Hardware.....	2
1.3.3 Orgware.....	2
1.4 Funktionalität.....	2
1.5 Daten.....	2
1.6 Benutzungsoberfläche.....	2
1.7 Qualitätsziele.....	2
1.8 Ergänzungen.....	2
2 Objektorientierte Analyse (OOA).....	3
2.1 Klassendiagramm.....	3
2.2 Use-Case-Diagramm.....	5
2.3 OOA Sequenzdiagramm.....	6
3 Objektorientierter Entwurf (OOD).....	8
3.1 Prototyp der Benutzeroberfläche.....	8
3.2 Paketdiagramm.....	13
3.3 OOD Sequenzdiagramm.....	14

4	Datenhaltung mittels einer relationalen Datenbank.....	15
4.1	Tabellenstruktur.....	15
4.2	Logische Schemas in SQL.....	15
	Quellenverzeichnis.....	17
	Anhang.....	18

Abbildungsverzeichnis

	Seite
Abbildung 1: Klassendiagramm zu den Klassen der Bergrettung.....	4
Abbildung 2: Use-Case-Diagramm.....	5
Abbildung 3: OOA Sequenzdiagramm.....	7
Abbildung 4: Startansicht der Software.....	9
Abbildung 5: Erfassungsfenster für einen Einsatz.....	9
Abbildung 6: Erfassungsfenster für Equipment.....	9
Abbildung 7: Erfassungsfenster für Personal.....	10
Abbildung 8: Erfassungsfenster für Patient.....	10
Abbildung 9: Ausgabe des „Read“-Buttons von Einsatz.....	11
Abbildung 10: Weiteres Interaktionselement.....	11
Abbildung 12: Fehlerprävention a.....	12
Abbildung 13: Fehlerprävention b.....	12
Abbildung 14: Fehlerprävention c.....	12
Abbildung 15: Paketdiagramm zur 3-Schichten Architektur.....	13
Abbildung 16: OOD Sequenzdiagramm zum Fallbeispiel Einfügen eines Patient.....	14
Abbildung 17: Klassendiagramm auf Tabellen.....	15

Abkürzungen

GUI	Graphische Benutzeroberfläche	(graphical user interface)
OOA	Objektorientierte Analyse	(object oriented analysis)
OOD	Objektorientierter Entwurf	(object oriented design)
UML	Unified Modeling Language	

1 Pflichtenheft

1.1 Zielbestimmung

1.1.1 Muss-Kriterien

- Zentrale Verwaltung von Daten, relevant für die Bergrettung (Equipment, Personal, Einsätze, Patienten)
- Daten in einer Datenbank einfügen, ändern und löschen
- Gespeicherte Daten sollen problemlos in einem *Graphical User Interface* abgerufen werden können
- Grafische Oberfläche soll alle benötigten Informationen auf Abruf bereithalten

1.1.2 Kann-Kriterien

keine

1.1.3 Abgrenzungskriterien

- Kein Zugriffsschutz (bei einer echten Anwendung wäre dies ein Muss-Kriterium, da jeder Prüfer nur für die eigenen Arbeitsthemen Zugriffe durchführen darf)
- Keine Zusammenfassung von Daten
- Keine grafische Analyse von Daten

1.2 Einsatz

1.2.1 Anwendungsbereiche

Verwaltungen von Rettungsdiensten, spezialisiert auf Gegenden mit Gebirge oder starkem Relief

1.2.2 Zielgruppen

- Einsatzkräfte
- Kostenstellen
- Versicherungen
- Krankenhäuser

1.2.3 Betriebsbedingungen

- Nutzung ausschließlich in Büroumgebung
- Datenbank muss stets aktuell gehalten werden
- Ständige Beobachtung nicht notwendig

1.3 Umgebung

1.3.1 Software

- Windows 8 / 10
- Java Development Kit 8 als Entwicklungsumgebung
- NetBeans Version 8.2

1.3.2 Hardware

Beliebiger Windows-Computer

1.3.3 Orgware

Einfügen von anfangs benötigten Daten (Personal, Equipment)

1.4 Funktionalität

Die Software ist auf das Verwalten und Erfassen anfallender Daten bei Bergrettungseinsätzen spezialisiert. Wobei auch präventive Maßnahmen, wie Baumfällarbeiten oder Reparaturarbeiten an hochalpinen Wegen, ohne Patientendaten aufgelistet werden können. Gesammelte Daten, werden zuverlässig in die Datenbank übertragen. Dort können Sie jederzeit abgeändert, gelöscht oder, bei Bedarf einfach, ausgelesen werden.

1.5 Daten

Durch Bergrettungseinsätze entstehende Daten sollen verwaltet werden. D.h. Daten, von beteiligtem Personal, involvierten Patienten, eingesetztem Equipment und den Einsatzeckdaten, werden zuverlässig in die Datenbank übertragen.

1.6 Benutzungsoberfläche

Es wird eine objektorientierte Oberfläche mit Menüs entsprechend der Gestaltungsvorschriften in den Lehreinheiten 9 und 10 erstellt.

1.7 Qualitätsziele

- Hohe Benutzerfreundlichkeit
- Flexible Abänderbarkeit
- Schnelle Übersicht

1.8 Ergänzungen

- Softwareentwicklungsprojekt wurde während der Ausbildung gefertigt

2 Objektorientierte Analyse (OOA)

Die OOA des vorliegenden Projekts „Bergrettung“ ist eine detailliertere Ausführung des Pflichtenhefts. Es beinhaltet sowohl das statische als auch das dynamische Modell. Das Klassendiagramm stellt die sechs Schritte des statischen Modells dar. Zu den Schritten gehört das Identifizieren der einzelnen Klassen, der Assoziationen, der Attribute, der Generalisierungsstrukturen, das Vervollständigen der Assoziationen, als auch das vollständige Spezifizieren der Attribute (vgl. Balzert 2005, S.155). Das dynamische Modell zeigt Funktionsabläufe und wird durch eine Use-Case-Schablone repräsentiert, die das zugehörige Use-Case-Diagramm genauer beschreibt (vgl. Balzert, S.11). Optional enthält das OOA-Modell des Projekts „Bergrettung“ ein Sequenzdiagramm, dass die Kommunikation zwischen allen wichtigen Akteuren eines Rettungseinsatzes abbildet.

2.1 Klassendiagramm

Ein Klassendiagramm ist ein Strukturdiagramm der UML, welches die Klassen an sich, die Generalisierung und die Assoziationen zwischen den Klassen darstellt. Klassendiagramme sind der beste Weg, um die Struktur eines zu entwerfenden oder abzubilden- den Systems, seine Attribute, deren Operationen sowie deren Beziehungen auf dieser Ebene in allen Details darzustellen (vgl. Balzert 2005, S.535).

Das folgende Klassendiagramm geht in erster Linie aus dem Punkt „Daten“ des Pflichtenhefts hervor. Dieser beschreibt die Art der zu verwaltenden Daten. Dazu gehören die persönlichen Daten des Personals, der Patienten, aber auch Equipment und allgemeine Eckdaten der Einsätze. Um das Programm denn eigenen Programmierfähigkeiten und den Anforderungen an das Verwaltungssystem anzupassen, besteht das Klassendiagramm aus vier Klassen. Jeder zu verwaltenden Datensatz bekommt eine eigene Klasse (Personal, Patient, Equipment und Einsatz) die individuell ausgearbeitet wird. Setter- und Getter-Methoden wurden bei diesem Diagramm absichtlich weggelassen, um Platz in der Dokumentation zu sparen. Augenmerk liegt darauf, dass nur unbedingt notwendige Daten der Bergrettung verwaltet werden können.

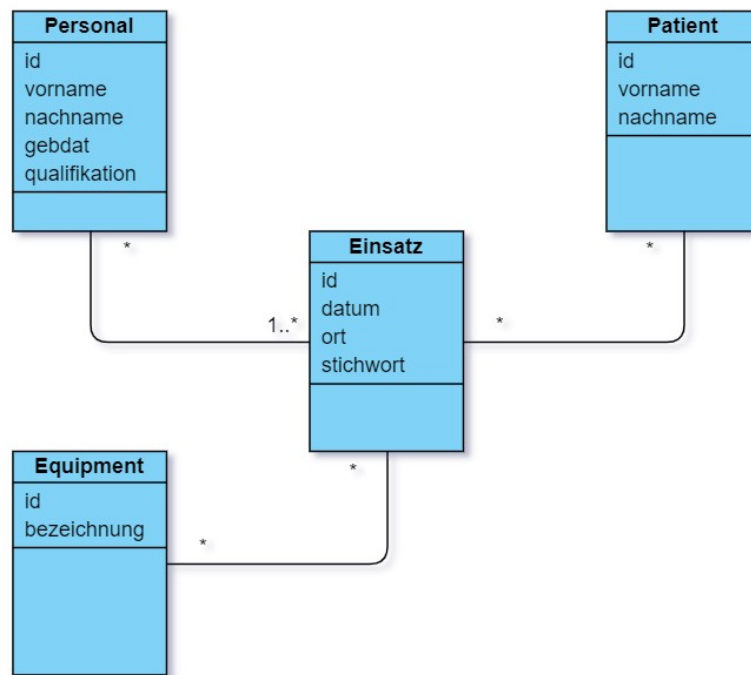


Abbildung 1: Klassendiagramm zu den Klassen der Bergrettung

Aus erstelltem Klassendiagramm (siehe Abbildung 1) lassen sich die Relationen Personal, Einsatz, Patient und Equipment, sowie deren entsprechende Koppeltabellen ableiten. Es kommt zu Koppeltabellen, da die Multiplizität im Klassendiagramm mit `*` als unbeschränkt beschrieben wird (vgl. Balzert 2005, S.43). Einzige Ausnahme ist, dass bei einem Einsatz immer eine Person des Personals vorhanden sein muss. Somit liegt die untere Grenze bei eins. Hierbei beschränken sich die Tabellen auf die Datentypen String, Integer und Date. Der Datentyp Integer wird ausschließlich für die IDs verwendet. String wird für alle restlichen Angaben verwendet.

Personal

id : Integer
 vorname : String
 nachname : String
 gebdat : String
 qualifikation : String

Einsatz

id : Integer
 datum : String
 ort : String
 stichwort : String

Patient

id : Integer
 vorname : String
 nachname : String

Equipment

id : Integer
 bezeichnung : String

2.2 Use-Case-Diagramm

Ein Use-Case-Diagramm beschreibt die Beziehung zwischen Akteuren und Use-Cases in einem System. Außerdem gibt ein Use-Case-Diagramm einen guten Überblick über das System und seine Schnittstellen zur Umgebung. Nebenbei werden durch das Diagramm die Use-Cases der durchzuführenden Aufgaben auf einem sehr hohen Abstraktionsniveau abgebildet (vgl. Balzert 2005, S.11).

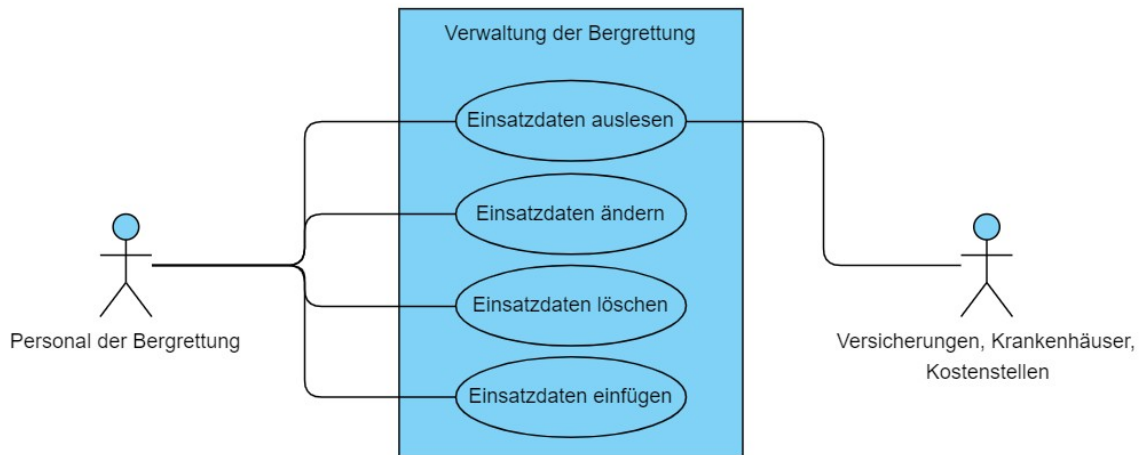


Abbildung 2: Use-Case-Diagramm

Dargestellt werden alle Instanzen, die auf die Datenbank der Bergrettungsverwaltung Zugriff haben. Akteure sind das Personal der Bergrettung, welche auf folgenden vier Use-Cases zugreifen können: Einsatzdaten ändern, Einsatzdaten löschen, Einsatzdaten einfügen und Einsatzdaten auslesen. Wobei Verwaltungsstellen von Krankenhäusern, Versicherungen und Kostenstellen nur über den Use-Case Einsatzdaten auslesen auf benötigte Daten zugreifen können. Auf die Use-Cases Einsatzdaten ändern, Einsatzdaten löschen und Einsatzdaten einfügen haben besagte Instanzen jedoch keinen Zugriff.

Ziel: Einfügen neuer Einsatzdaten, Abändern bereits gespeicherter Datensätze, Auslesen benötigter Einsatzdaten und Löschen von Einsatzdaten

Kategorie: primär

Vorbedingung: -

Nachbedingung Erfolg:

korrekte Datensätze zur weiteren Verarbeitung in der Datenbank

Nachbedingung Fehlschlag:

Datenbank der Bergrettung fehlerhaft, falsche Einsatzdaten, eventuelle Anomalien der Datenbank

Akteur: Personal, Versicherungen, Krankenhäuser, Kostenstellen

Auslösendes Ereignis: Unfall oder präventive Arbeiten

Beschreibung:

- 1 Kontaktaufnahme mit der Bergrettung
- 2 Prüfen welche Art und Ausmaß von Ereignis vorliegt
- 3 Prüfen der Zuständigkeit
- 4 Zusammenstellen des einsatzfähigen Personals, zu einem Einsatzteam
- 5 Zusammenstellen des Benötigten Equipments, je nach Art des Ereignisses
- 6 Ausrücken zum Einsatzort

Erweiterung:

- 2 Art und/oder Ausmaß sind nicht bekannt. Verständigen weiterer Institutionen, wie Feuerwehr und Polizei.
- 3 Zuständigkeit liegt bei einer anderen Stelle, kein Einsatz der Bergrettung. Somit auch kein neuer Datensatz für die Datenbank der Verwaltung.
- 4a Kein einsatzfähiges Personal. Kontaktieren anderer Institutionen.
- 4b Zu wenig einsatzfähiges Personal. Anfordern von Verstärkung anderer Institutionen.
- 5 Kein passendes Equipment für die Art des Ereignisses. Ausleihen von einer anderen Bergrettung bzw. anderer Institution die passendes Equipment besitzt.

Alternativen: -

(vgl. Balzert 2005, S.138)

2.3 OOA Sequenzdiagramm

Mit Hilfe des Sequenzdiagramms, einer Unterart des Interaktionsdiagramms, werden Szenarien modelliert. Sequenzdiagramme stellen dabei graphisch das Zusammenspiel der Kommunikationspartner dar (vgl. Balzert 2005, S.81). Ausgangspunkt des Sequenzdiagramms ist das Use-Case-Diagramm.

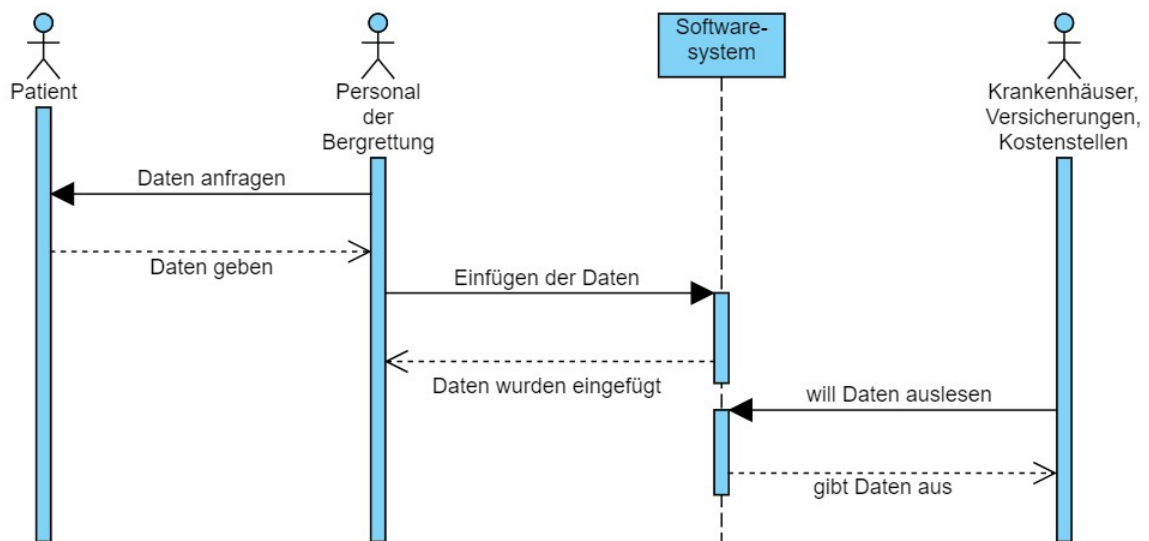


Abbildung 3: OOA Sequenzdiagramm

Im folgenden Sequenzdiagramm (siehe Abbildung 3) wird die Kommunikation zwischen Patienten, Personal, dem Softwaresystem und den verwaltungstechnischen Instanzen beschrieben. Als Beispiel wird ein Datensatz eines Patienten mit id, vorname, nachname an das Personal der Bergrettung übergeben. Dieses fügt die Daten des Patienten der Verwaltungssoftware hinzu. Auf das Softwaresystem können Instanzen, wie Krankenhaus und Kostenstelle zugreifen, die die persönlichen Daten und die Umstände des Unfalls benötigen. Jedoch ist der Zugriff für die einzelnen auf das Auslesen von Datensätzen beschränkt. Das Sequenzdiagramm zeigt für die Aktionen des Personals exemplarisch nur das Einfügen. Unter anderem stehen dem Personal der Bergrettung, neben dem Einfügen eines Einsatzdatensatzes, auch andere Use-Cases wie das Auslesen, Ändern und Löschen von Einsatzdatensätzen zur Verfügung. Instanzen können nur mit dem Softwaresystem interagieren, während das Personal der Bergrettung in der Lage ist gleichzeitig mit den Patienten zu kommunizieren und auf das System zuzugreifen. Der Patient hingegen hat nur die Möglichkeit mit dem Personal der Bergrettung zu interagieren. In Abbildung 3 wird das erfolgreiche Einfügen eines neuen Patienten seitens des Personals der Bergrettung, als auch das erfolgreiche Auslesen seitens der Instanzen, dargestellt. Somit liegen Szenarien vor, die eine erfolgreiche Bearbeitung beschreiben (vgl. Balzert, S.80).

3 Objektorientierter Entwurf (OOD)

Grundlage des OOD-Modells ist das OOA-Modell; es enthält alle detaillierten Informationen, aus denen sich der Prototyp der Benutzeroberfläche und die Realisierung bzw. Implementierung ableiten lassen (vgl. Balzert 2005, S.11). Ergebnisse des OOD-Modells sind die Spezifikation der Klassen aus Sicht der Realisierung und die vollständige Erstellung der Softwarearchitektur auf einem hohen Abstraktionsniveau (vgl. Balzert 2005, S.539). Das OOD des Projekts „Bergrettung“ umfasst eine genau Beschreibung des Prototyps der Benutzeroberfläche, ein Paketdiagramm, das die 3-Schichten Architektur der Software darstellt und ein Sequenzdiagramm zum Fallbeispiel „Einfügen eines Patienten“.

3.1 Prototyp der Benutzeroberfläche

„Der Prototyp der Benutzeroberfläche ist ein ablauffähiges Programm, dass alle Attribute des OOA-Modells auf die Oberfläche abbildet“ (Balzert 2005, S.12). Der Prototyp dient dazu zusammen mit dem Benutzer das bereits bestehende OOA-Modell zu begutachten und eventuell zu verbessern, falls nötig. Dabei besteht dieser jedoch nur aus Fenstern und realisiert keinerlei Anwendungsfunktionen. (vgl. Balzert, S.12). Die GUI orientiert sich sehr stark am Klassendiagramm des OOA-Modells. Es zeigt die am Entwurfsmuster beteiligten Klassen, demnach wurden die vier Klassen (Personal, Patient, Einsatz und Equipment) als eigenständige Erfassungsfenster realisiert

Um dem Personal, als auch den Verwaltungsstellen der einzelnen Instanzen (Krankenhäusern, Versicherungen, Kostenstellen, etc.), einen möglichst simplen Zugriff auf die Daten der Bergrettungsverwaltung zu gewährleisten, wurde die Benutzeroberfläche einfach verständlich und leicht bedienbar designet. Klar im Vordergrund steht, dass die Benutzerfreundlichkeit der Funktionalität in nichts nachsteht. Zu betonen ist jedoch, dass es sich hierbei um einen Prototyp handelt, eine grobe Grundform, auf der alle weiteren Verfeinerungen (virtuelle Linien, einheitliche Größen der Button, etc.) und Veränderungen der Benutzeroberfläche aufbauen.

Die Benutzeroberfläche beginnt mit der Startansicht auf der die vier Button „Einsatz“, „Equipment“, „Patient“ und „Personal“ zu sehen sind. Um einen Vorgang abzubrechen steht jederzeit der X-Button in der rechten, oberen Ecke zur Verfügung.

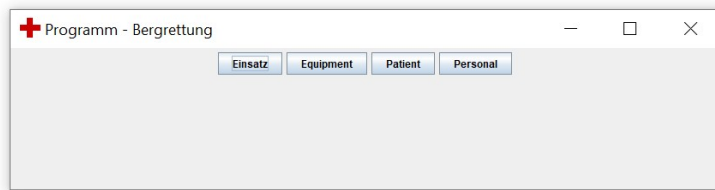


Abbildung 4: Startansicht der Software

Bei der grundlegenden Gestaltung der Fenster wurde auf Einheitlichkeit geachtet. Einzeilige Textfelder bzw. Eingabefelder (vgl. Balzert, S.239), sowie auf eindeutig beschriftete Schaltflächen (vgl. Balzert, S. 240).

Nachdem drücken des „Einsatz“-Buttons der Startansicht:

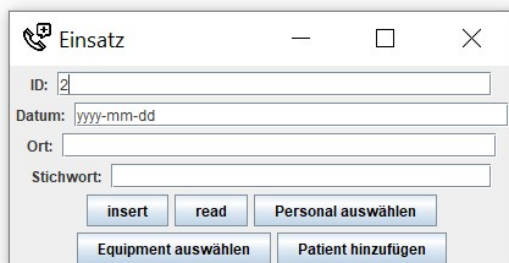


Abbildung 5: Erfassungsfenster für einen Einsatz

Mit dem Erfassungsfenster wird ein neuer Einsatz durch „insert“ angelegt, dafür benötigt das Programm alle vier Attribute (ID, Datum, Ort und Stichwort). Zum Einsatz an sich kann man benötigtes Equipment, Personal und die/den Patienten hinzufügen. Zum Auslesen mit „read“ reicht die ID des bereits gespeicherten Einsatzes aus, es ist jedoch nur das Auslesen eines einzigen Einsatzes möglich. Um dies zu tun werden die entsprechenden Erfassungsfenster mit Hilfe der Button „Equipment auswählen“, „Personal auswählen“ und „Patient hinzufügen“ aufgerufen.

Nachdem drücken des „Equipment“-Buttons der Startansicht:

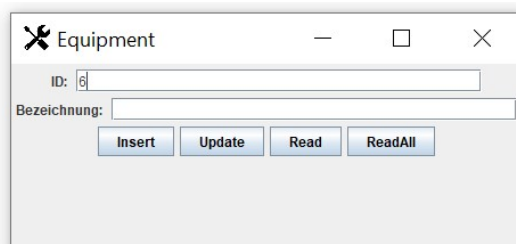


Abbildung 6: Erfassungsfenster für Equipment

Der Aufbau des Erfassungsfenster für Equipment (siehe Abbildung 6) ist dem von Abbildung 5 sehr ähnlich. Einzige Unterschiede sind die Buttons „Update“ und „ReadAll“. Der Button „Update“ benötigt eine bereits vorhandene ID zusammen mit einer neuen Bezeichnung. Der Button „ReadAll“ benötigt keinerlei Eingaben, da er alle gespeicherten Geräte und Fahrzeuge aus Equipment ausgibt.

Nachdem Drücken des „Personal“-Buttons der Startansicht:

The screenshot shows a window titled 'Personal' with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there are several input fields: 'ID:' with a text box containing the number '5'; 'Nachname:' with an empty text box; 'Vorname:' with an empty text box; '(Geburtstag:)' with a text box containing 'yyyy-mm-dd'; and '(Qualifikation:)' with an empty text box. Below these fields are four buttons: 'Insert', 'Update', 'Read', and 'ReadAll'.

Abbildung 7: Erfassungsfenster für Personal

Die Funktion der Button mit identischem Namen bleibt bei jedem Erfassungsfenster gleich. Beispielsweise funktioniert der „ReadAll“-Button bei Abbildung 5 genauso wie bei Abbildung 7. Das Erfassungsfenster von Personal (siehe Abbildung 7) ist das Einzige mit optionalen Attributen, welche in Klammern stehen. Sie können angegeben werden, müssen aber nicht. Farblich soll der Unterschied zwischen obligatorischen und optionalen Eingaben noch durch eine unterschiedliche Untergrundtönung der Eingabefelder hervorgehoben werden (vgl. Balzert, S.239). Derzeit sind optionale Eingaben durch eine in Klammern gesetzte Beschriftung gekennzeichnet (siehe Abbildung 7: Geburtstag und Qualifikation).

Nachdem Drücken des „Patient“-Buttons des Startansicht:

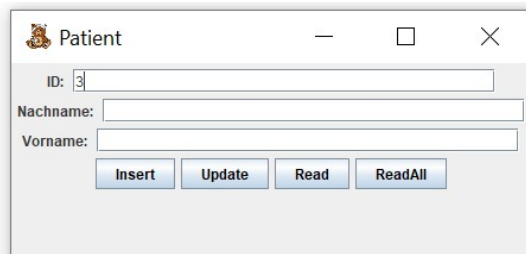
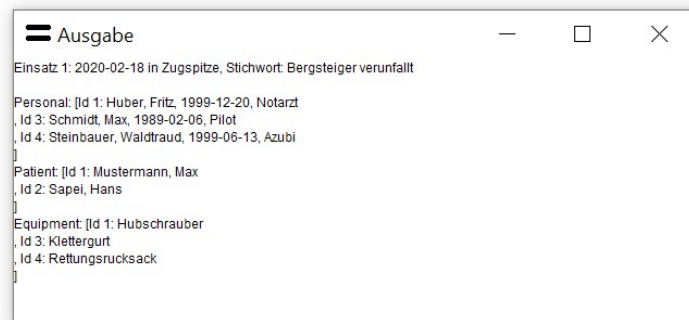
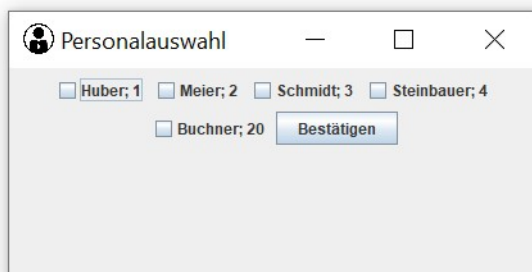
The screenshot shows a window titled 'Patient' with a standard Windows-style title bar. Inside the window, there are three input fields: 'ID:' with a text box containing the number '3'; 'Nachname:' with an empty text box; and 'Vorname:' with an empty text box. Below these fields are four buttons: 'Insert', 'Update', 'Read', and 'ReadAll'.

Abbildung 8: Erfassungsfenster für Patient

Nachdem Drücken des „Read“-Buttons des Erfassungsfenster Einsatz

**Abbildung 9: Ausgabe des „Read“-Buttons von Einsatz**

In der Abbildung 9 wird der Einsatz mit der ID 1 ausgegeben. Zu sehen ist: Wann der Einsatz stattgefunden hat, eine grobe Umschreibung was passiert ist, welches Personal dabei war, die Daten des Patienten und das verwendete Equipment. Das verwendete Interaktionselement wird mehrzeiliges Textfeld genannt, in diesem Projekt dient es zur reinen Informationsausgabe und ist für den Benutzer schreibgeschützt. Testausgaben werden linkbündig ausgegeben. Da die Größe des mehrzeiligen Testfeldes für die Ausgabe der Informationen eines Einsatzes vollkommen ausreichend ist, wurde für das Projekt, an dieser Stelle auf Rollbalken verzichtet (vgl. Balzert 2005, S.240).

**Abbildung 10: Weiteres Interaktionselement**

Eine weiteres Interaktionselement, das Verwendung gefunden hat, ist das Kontrollkästchen, bei dem eine Mehrfachauswahl an Auswahlmöglichkeiten getroffen werden kann. Anders wie bei dem Optionsfeld, dass nur für eine Einfachauswahl geeignet ist (vgl. Balzert 2005, S.240). In Abbildung 10 wird das Personal zeilenweise mit einem Abstand von drei Zeichen angeordnet; dies soll jedoch für die fertige Benutzeroberfläche in eine vertikale Anordnung abgeändert werden (vgl. Balzert 2005, S. 241).

Die GUI ist so designt, dass sie ihren Benutzer bei der Bedienung unterstützen kann. Indem das ID-Textfeld bei Aufrufen eines Erfassungsfensters mit der nächsten, freien ID vorausgefüllt wird oder der Benutzer auf Fehler bei der Eingabe hingewiesen wird.

Die für dieses Projekt entwickelte GUI kann drei fehlerhafte Eingaben des Benutzers präventiv abfangen und so inkorrekte oder unvollständige Dateneingaben verhindern. Zu den Fehlern gehören falsche Eingaben (siehe Abbildung 14), unvollständige Eingaben (siehe Abbildung 12) und das Aufmerksam machen auf bereits vorhandene Datensätze (siehe Abbildung 13), welche durch das Prüfen der ID ermittelt werden. Um den Benutzer darauf aufmerksam zu machen, färbt sich das fehlerhafte Textfeld rot ein und der Benutzer ist nicht in der Lage seinen derzeitigen Vorgang weiter auszuführen, bevor der Fehler nicht behoben wurde.

The screenshot shows a window titled 'Personal' with a user icon. It contains several input fields: 'ID:' with the value '5', 'Nachname:' (redacted with a red bar), 'Vorname:' (redacted with a red bar), '(Geburtsdag:)' with the value '2020-02-20', and '(Qualifikation:)' with the value 'Hubschrauberpilot'. At the bottom, there are four buttons: 'Insert', 'Update', 'Read', and 'ReadAll'.

Abbildung 11: Fehlerprävention a

Notwendige Daten wurden vergessen einzutragen.

The screenshot shows a window titled 'Patient' with a patient icon. It contains input fields: 'ID:' (redacted with a red bar), 'Nachname:' with the value 'Steinbauer', and 'Vorname:' with the value 'Isabel'. At the bottom, there are four buttons: 'Insert', 'Update', 'Read', and 'ReadAll'.

Abbildung 12: Fehlerprävention b

Datensatz ist bereits vorhanden.

The screenshot shows a window titled 'Equipment' with a wrench icon. It contains input fields: 'ID:' with the value 'Isabel' (redacted with a red bar), and 'Bezeichnung:' with the value 'Hubschrauber'. At the bottom, there are four buttons: 'Insert', 'Update', 'Read', and 'ReadAll'.

Abbildung 13: Fehlerprävention c

In Abbildung 14 wurde ein falscher Datensatz eingegeben. Werden nämlich nicht numerischen Eingaben für das Attribut ID getätigt, so werden diese abgewiesen. Die fehlerhafte Zeile wird rot eingefärbt. Weitere Funktionalitäten, die normalerweise mit den Buttons aufgerufen werden, sind solange gesperrt bis die entsprechenden Fehler behoben sind.

3.2 Paketdiagramm

„Paketdiagramme dienen zur Darstellung von Paketen, Paketverschmelzungen, Paketimporten und Abhängigkeitsbeziehungen.“ (Balzert 2005, S.309). Bei dem Projekt „Bergrettung“ werden mit Hilfe des Paketdiagramms verschiedene Architekturschichten, wie graphische Benutzeroberfläche, Datenbankzugriff, etc. dargestellt und präzise voneinander getrennt (vgl. Balzert 2005, S.540).

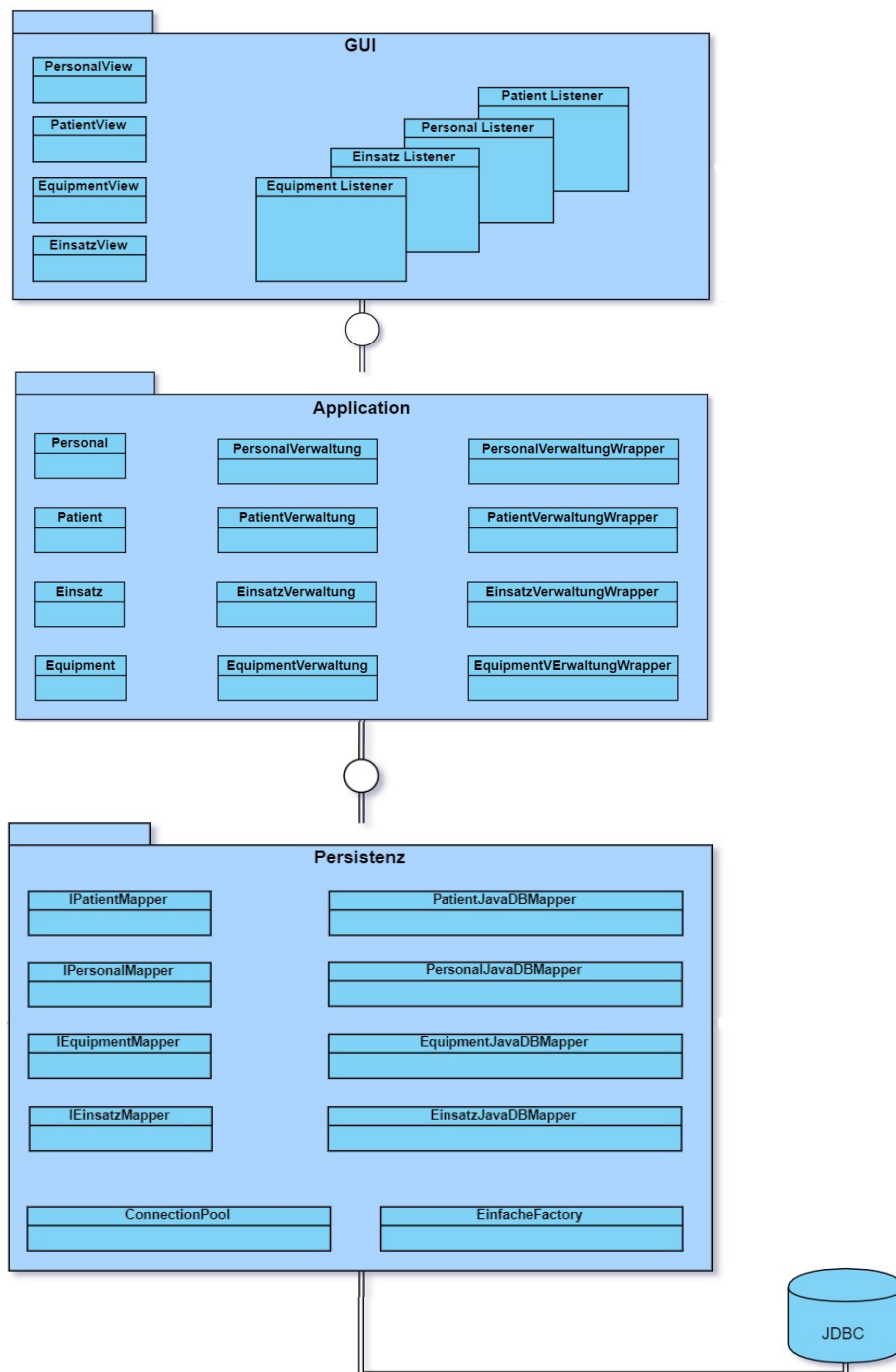


Abbildung 14: Paketdiagramm zur 3-Schichten Architektur

Das Softwaresystem ist in drei Schichten unterteilt, die Persistenz, die Applikation und das Graphical User Interface. Jede dieser Schichten hat seine eigene Aufgabe und Kommunikationsschnittstellen zu den jeweils anderen Schichten.

Die Persistenz ist die Schnittstelle des Programms, um auf die Datenbank zugreifen zu können. Sie fügt Einträge hinzu, ändert vorhandene Einträge und gibt diese aus.

Die Applikation ist der Verwaltungslayer, der die Eingaben aus der graphischen Benutzeroberfläche an die Persistenzschicht weiterleitet. Hierzu wird zusätzlich eine Wrapperklasse verwendet, damit die Verwaltung nicht direkt auf die Persistenz zugreifen muss. Das Graphical User Interface ist die Bedienoberfläche, die dem Benutzer zur Verfügung gestellt wird. Hier gibt es zum einen die Fenster (Views), welche der Benutzer sieht und zum anderen die ActionListener, welche bei Betätigen der verschiedenste Button diverse Aktionen ausführen.

3.3 OOD Sequenzdiagramm

Das OOD Sequenzdiagramm stellt graphisch die Kommunikation innerhalb des Systems dar, enthalten ist hierbei die Anbindung des Softwaresystems an die Datenbank. Für den Kunden unrelevante fachliche Lösungen wie das System agiert, werden hier aufgeführt. Im Beispiel wird ein neuer Patient in die Datenbank eingefügt.

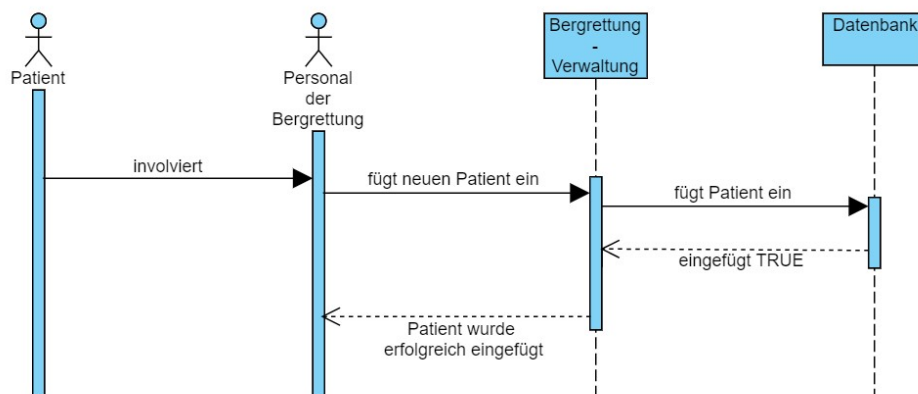


Abbildung 15: OOD Sequenzdiagramm zum Fallbeispiel Einfügen eines Patient

4 Datenhaltung mittels einer relationalen Datenbank

4.1 Tabellenstruktur

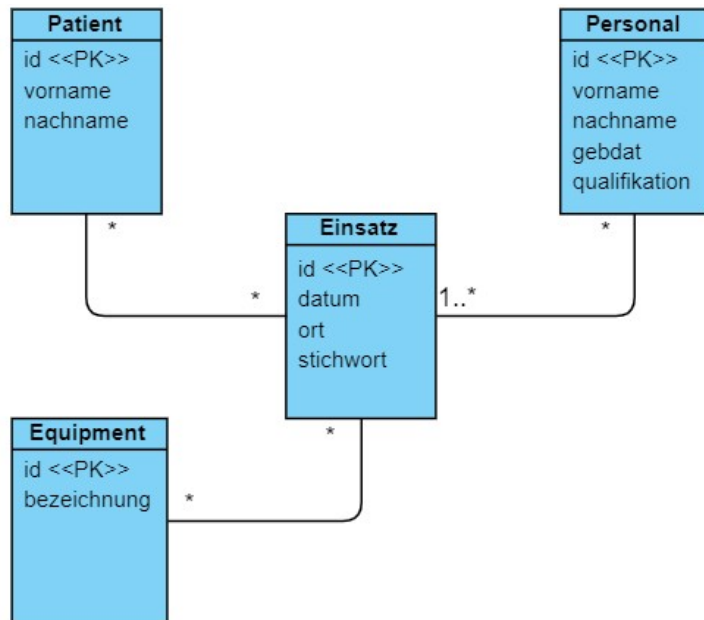


Abbildung 16: Klassendiagramm auf Tabellen

4.2 Logische Schemas in SQL

Das System ist so programmiert, dass die Tabellen selbstständig erzeugt werden.

Tabellen:

```
create table einsatz
( id int primary key,
  datum date,
  ort varchar (20),
  stichwort varchar (40) );
```

```
create table equipment
( id int primary key,
  bezeichnung varchar (40) );
```

```
create table personal
( id int primary key,
  vorname varchar (20),
  nachname varchar (20),
  gebdat date,
  qual varchar (40) );
```

```
create table patient
( id int primary key,
  vorname varchar (20),
  nachname varchar (20) );
```

Koppeltabellen:

```
create table einsatz_equipment  
( eqid int, eid int, primary key (eqid, eid) );
```

```
create table einsatz_patient  
(paid int, eid int, primary key (paid, eid) );
```

```
create table einsatz_personal  
(pid int eid int, primary key (pid, eid) );
```

Quellenverzeichnis

Literaturquellen

„Balzert, Hilde [2005]: Lehrbuch der Objektmodellierung – Analyse und Entwurf mit der UML 2, Heidelberg, 2. Auflage.“

Bildquellen

Rotes Kreuz: <https://pixnio.com/free-images/2017/12/03/2017-12-03-17-25-26-927x900.png>

Personal Bild: <https://perfect-illusions.de/wp-content/uploads/personalicon.png>

Einsatz (Telefon): https://www.flaticon.com/free-icon/emergency-call_124992

Werkzeug: <https://image.flaticon.com/icons/png/512/100/100852.png>

Ergebnis: https://cdn.icon-icons.com/icons2/1144/PNG/512/equalsign_81021.png

Anhang

Test der GUI

Der erste Teil des Tests besteht aus dem sogenannten Blackbox Test. Bei diesem wird versucht, nicht funktionierende Methoden/Klassen zu finden. Dazu werden Testklassen erstellt, welche die Methoden der zu testenden Klasse aufrufen und mit dem Sollwert vergleichen. So werden Methoden gefunden, die Daten nicht korrekt speichern, auslesen oder bearbeiten. Das Programm im Hintergrund interessiert erst einmal nicht. In dem Projekt sind deshalb exemplarisch für jede Schicht ein paar Testklassen erstellt worden, die beispielsweise überprüfen, ob ein Objekt der Klasse Personal seine Attribute korrekt ausgibt und ändert. Des Weiteren ist auf Grund der Fehleranfälligkeit der Persistenz-Schicht für jede Mapper-Klasse ein Test vorhanden, die alle Datenbankzugriffe testet. Schlägt der Test einer Methode/Klasse fehl so wird in der Konsole der Methodenname rot ausgegeben. Ist der Test für eine Methode erfolgreich, wird der ein Text mit dem Methodenname und dem Wort erfolgreich in der Konsole ausgegeben.

Zuerst wurden alle Schaltflächen getestet, ob diese die richtigen Fenster öffnen bzw. überhaupt ein Fenster öffnen. Haben sie die gewünschte Größe? Position? Danach wurden die Eingabefelder ebenfalls auf ihre gewünschte Funktionalität überprüft. Zeigen sie Eingabefehler an und färben sich rot? Übernehmen Sie die eingegebenen Daten und geben diese auch wieder korrekt aus? Befinden sie sich an der richtigen Stelle im Fenster?

Whitebox-Test

Durch ausreichende Schnittstellen zu den einzelnen Schichten des Systems, wird diese leicht abänderbar und somit leicht zu warten. Die Namenskonventionen wurden alle von der Gruppe eingehalten. Auch vorhandene Modelle auf einem hohem Abstraktionsniveau sind leicht verständlich und spiegeln das Softwaresystem in ihrer Gesamtheit wieder. Jede Komponente des Systems hat ihre eigene Aufgabe, die auch erfüllt wird, demnach spricht man bei diesem System auch von einem stark gebundenen System.

Nichtfunktionale Anforderungen

Das vorliegende Programm ist leicht verständlich. Die Bedienung der GUI wird durch den Punkt ‚Prototyp der Benutzeroberfläche‘ der Dokumentation, genau erklärt, sodass bei Fragen diesbezüglich jederzeit Abhilfe geschaffen wird. Die GUI unterstützt ihren Benutzer bei der Bedienung des Programms und schützt ihn vor Fehleingaben, zeigt

ihm was nicht wie gewünscht funktioniert. Somit ist der Benutzer in der Lage sich ohne fremde Hilfe selber zu korrigieren. Das Programm an sich verfügt über eine hohe Nutzerfreundlichkeit, es läuft problemlos ohne Abstürze, für eine mehrfach Eintragung von Daten wird kein Neustart benötigt und durch die performante als auch schnelle Verständlichkeit, ist das Softwareprogramm das Beste Hilfsmittel bei der Verwaltung von Daten, die bei Bergrettungseinsätzen anfallen.