Hochschule Bremerhaven

Fachbereich II – Management und Informationssysteme

Informatik und Wirtschaftsinformatik

Modul SWE II

Fahrzeitenmanagement System

von

Alina Hoeborn Matrikel-Nr. 40866
Fabian Tober Matrikel-Nr. 40607
Laura Halbeck Matrikel-Nr. 37578

Team05 - F.A.L.B

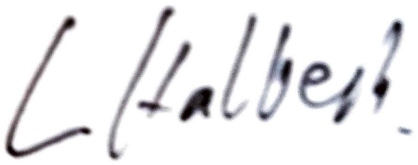
01.08.2024

Inhaltsverzeichnis

1 Eigenständigkeitserklärung

Hiermit erklären wir, dass wir die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.
Alle sinngemäß und wörtlich übernommenen Textstellen aus fremden Quellen wurden kenntlich gemacht.

Bremerhaven, den 14.07.2024
Alina Hoeborn, Fabian Tober, Laura Halbeck



2 Überblick und Vorgehensbeschreibung des Gesamtprojektes

Erstellt von: Fabian

2.1 Einleitung

Das Projekt "Fahrzeitenmanagement System"wurde von der Fahrschule Frank Martens in Bremerhaven in Auftrag gegeben. Ziel dieses Projekts ist es, ein digitales System zu entwickeln, das es Fahrschülern ermöglicht, Fahrtermine zu buchen und ihre bereits absolvierten Theorie- und Praxisstunden einzusehen. Zusätzlich soll das System Informationen über Theoriestunden bereitstellen, einschließlich der nächsten Termine und Orte, sowie eine Information zur Anmeldung für neue Fahrschüler bieten.

2.2 Projektziel

Das Hauptziel des Projekts ist die Entwicklung eines digitalen Systems zur Verwaltung und Optimierung der Fahrzeitenplanung und -buchung. Dies soll dazu beitragen, organisatorische Aufgaben zu reduzieren und den Fahrschülern einen Einblick in ihren Fortschritt zu ermöglichen.

2.3 Hintergrund und Kontext

Derzeit verwendet die Fahrschule Frank Martens ein System, das den Fahrschülern keinen Einblick in ihren Buchungen und Fortschritten bietet, was oft zu Unübersichtlichkeit führt. Das neue Fahrzeitenmanagement System soll sowohl Fahrlehrern als auch Fahrschülern eine einfache Verwaltung und Einsicht in Termine und Fortschritte ermöglichen und somit den gesamten Ablauf optimieren.

2.4 Projektumfang

Das System umfasst die Planung, Buchung und Verwaltung von Fahrstunden, die Zuweisung von Fahrlehrern sowie die Verwaltung von Schülerprofilen und Ausbildungsfortschritten. Nicht enthalten sind die Abwicklung von Zahlungen und personenbezogene Daten, mit Ausnahme von Namen der Fahrschüler und Fahrlehrer.

2.5 Erwartete Ergebnisse

Zu den erwarteten Ergebnissen des Projekts gehören:

- Ein digitales Buchungssystem für Fahrstunden
- Eine (automatisierte) Zuweisung von Fahrlehrern
- Ein Web-Portal für Fahrlehrer zur Verwaltung ihres Kalenders und der Schülerfortschritte
- Zugang für Fahrschüler, der es ihnen ermöglicht, Praxisstunden zu buchen und ihren Fortschritt einzusehen
- Integration von Informationen zu Theoriestunden, einschließlich Zeiten und Standorten

2.6 Projektziele und Anforderungen

Zu den Anforderungen an das System zählen:

- Eine benutzerfreundliche Oberfläche zur Buchung von Fahrstunden
- Ein Echtzeit-Kalender mit Verfügbarkeiten der Fahrlehrer und Theorie-, sowie Praxisstunden
- Ein Buchungssystem für Praxisstunden
- Informationen über den Fortschritt der Fahrschüler

2.7 Vorgehensweise

Nach ausführlichen Gesprächen mit der Fahrschule über die gewünschten Funktionen haben wir die Planungsphase eingeleitet und einen groben Projektplan mittels UML-Diagrammen erstellt. Dies diente dazu, den genauen Bedarf zu ermitteln.

Im Anschluss an die Planung begannen wir mit der Implementierung der besprochenen Funktionen und starteten die Entwicklung der Webseite. Dabei überprüften wir die Umsetzbarkeit der geplanten Features und nahmen gegebenenfalls Anpassungen vor. Zu diesem Zeitpunkt konnten jedoch noch nicht alle Funktionen getestet werden.

Parallel zur grundlegenden Implementierung begannen wir mit der Erstellung der Dokumentation, einschließlich Spezifikationen und Berichten, da es sich bei der Webseite nur um einen Prototyp handelt.

3 Projektorganisation und Teamarbeit

Erstellt von: Fabian

3.1 Einleitung

Da ein Fahrzeitenemanagement System ein komplexeres Projekt darstellt haben wir zu Beginn ein grundlegendes Konzept entwickelt, um eine solide Grundlage für unsere Arbeit zu schaffen.

3.2 Zieldefinition

Um effektiv arbeiten zu können, haben wir als Erstes unsere Ziele klar definiert. Wir haben unser Hauptziel in mehrere zwischen Ziele aufgeteilt damit das Arbeiten einfacher fällt. Das Hauptziel war die Entwicklung eines digitalen Systems zur Verwaltung und Optimierung der Fahrzeitenplanung und -buchung.

3.3 Projektplanung und Struktur

Wir haben einen Projektleiter ernannt und die Aufgaben im Team grob aufgeteilt, damit jeder wusste, was zu tun war. Es gab unregelmäßige Treffen und Diskussionen sowie gelegentliche Wechsel der Aufgabenbereiche, um Flexibilität zu gewährleisten. Hierbei war es uns wichtig nicht regelmäßige treffen zu vereinbaren, sondern eher flexible je nach Meilensteine da einige länger als andere brauchen und Probleme nicht regelmäßig auftreten. Obwohl nicht alle Meilensteine rechtzeitig erreicht wurden, haben wir alle Aufgaben rechtzeitig erledigt, da Verzögerungen im Voraus besprochen wurden.

3.4 Tools

Für Teamtreffen haben wir Discord verwendet, was uns eine unkomplizierte und zuverlässige Möglichkeit bot, uns zu versammeln und wichtige Themen zu besprechen. Für schriftliche Fragestellungen und Diskussionen nutzten wir einen geschützten Matrix-Raum, der problemlos funktionierte. Zur allgemeinen Organisation und Festlegung von Meilensteinen setzten wir OpenProject ein. Obwohl wir das volle Potenzial dieser Plattform nicht ausgeschöpft haben, half sie uns dennoch dabei, einen besseren Überblick über unsere Meilensteine zu behalten. Git wurde verwendet, um den aktuellen Code zu verwalten, was Duplikationen vermied und die Nachverfolgung von Problemen erleichterte sowie die Rückkehr zu früheren Versionen ermöglichte, wenn nötig.
Insgesamt war unsere Projektorganisation effektiv und die gut koordinierte Teamarbeit trug wesentlich zum erfolgreichen Abschluss unseres Projekts bei.

4 Team

Erstellt von: Fabian

Hier gehen wir nun genauer auf das Team als Überblick ein.

4.1 Teamorganisation

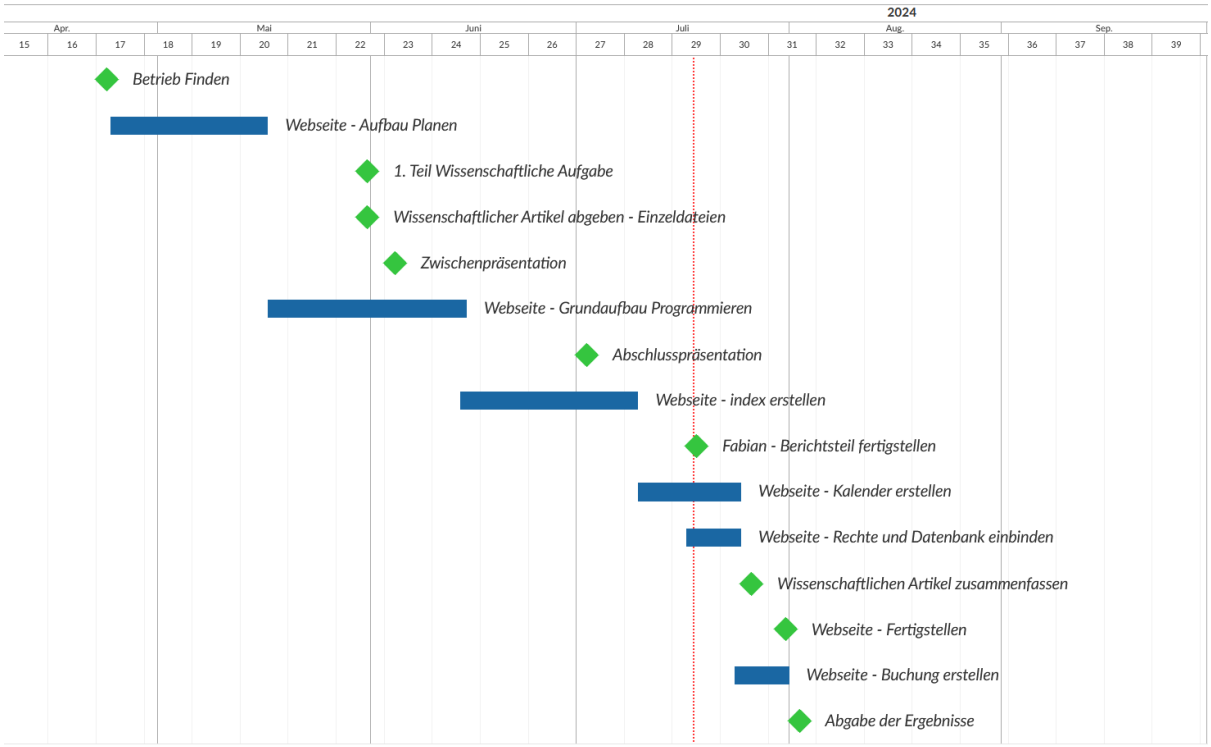
Name	Aufgabe
Alina Hoeborn	Bericht, Wissenschaftlicher Artikel, UML, Kundenkontakt
Fabian Tober	Bericht, Wissenschaftlicher Artikel, Webseite
Laura Halbeck	Bericht, Wissenschaftlicher Artikel, UML, Datenbank anbindung

4.2 Teamarbeit

Die Teamarbeit war durch offene Kommunikation geprägt. Unregelmäßige Teamtreffen wurden abgehalten, um alle auf dem aktuellen Stand zu halten, Probleme zu besprechen und Fragen zu klären.

5 Gantt-Diagramm
Erstellt von: Fabian

Folgend sehen wir das Gantt-Diagramm, das den Projektplan für die Entwicklung der Webseite und die damit verbundenen Aufgaben darstellt. Die Aufgaben sind in Form von Balken und Meilensteinen dargestellt, da wir viel mit Meilensteinen gearbeitet haben. Dieses Diagramm hat uns grundlegende Informationen geliefert, sodass wir nichts vergessen haben.



6 Analyse- und Erhebungsmethoden für die Anforderungen
Erstellt von: Fabian

6.1 Anforderungsermittlung

Um herauszufinden, was genau gewünscht ist, haben wir uns als Erstes mit der Unternehmensleitung zusammengesetzt und ihre Vorstellungen besprochen. Anschließend haben wir besprochen welche Systeme vorhanden sind und ob das “Fahrzeitenmanagement System” irgendwie an deren Systeme angepasst werden muss.

Nach der Anforderungsanalyse haben wir uns zu einem Brainstorming getroffen und erste UML-Diagramme erstellt, um ein besseres Verständnis und klare Vorstellungen zu entwickeln.

6.2 FURPS

Während unseres Brainstormings haben wir das „FURPS“-Modell entwickelt, um einen präzisen Plan zu erstellen und sicherzustellen, dass wir nicht vom Thema abweichen.

6.2.1 Funktionale Anforderungen

Wir haben definiert, welche Funktionen das System erfüllen soll:

- Ein digitales Buchungssystem für Fahrstunden
- Eine (automatisierte) Zuweisung von Fahrlehrern
- Ein Web-Portal für Fahrlehrer zur Verwaltung ihres Kalenders und der Schülerfortschritte
- Zugang für Fahrschüler, der es ihnen ermöglicht, Praxisstunden zu buchen und ihren Fortschritt einzusehen
- Integration von Informationen zu Theoriestunden, einschließlich Zeiten und Standorten

6.2.2 Benutzbarkeit

Wir haben festgelegt, wie benutzerfreundlich und intuitiv das System sein soll:

- Die Benutzeroberfläche soll intuitiv und einfach zu verstehen sein
- Der Login-Prozess soll unkompliziert und benutzerfreundlich gestaltet werden
- Kontaktmöglichkeiten sollen leicht auffindbar und zugänglich sein

6.2.3 Zuverlässigkeit

Wir haben spezifiziert, wie stabil und fehlerresistent das System sein soll:

- Das System soll auch bei Fehlern in einer Komponente (z. B. bei fehlerhaften Eingaben) weiterhin funktionsfähig bleiben
- Es sollen geplante Wartungszeiten festgelegt werden, um die Auswirkungen auf die Benutzer zu minimieren

6.2.4 Leistung

Wir haben definiert, welche Anforderungen an die Geschwindigkeit und Effizienz des Systems gestellt werden:

- Das System muss innerhalb von 2 Sekunden auf Benutzeraktionen reagieren
- Das System soll in der Lage sein, bis zu 50 gleichzeitige Benutzer zu unterstützen, ohne signifikante Leistungseinbußen zu erfahren

6.2.5 Wartbarkeit

Wir haben festgelegt, wie wartungsfreundlich und aktualisierbar das System sein soll:

- Das System erfordert keine regelmäßigen Wartungen oder Aktualisierungen, da es ausschließlich für die Speicherung und das Abrufen von Daten konzipiert ist.

7 Ergebnisse der Ist-Analyse

Erstellt von: Alina

Wir haben eine Ist-Analyse mit der Fahrschule durchgeführt, um herauszufinden bei welchen Prozessen es ineffiziente Abläufe gibt, die wir so gut wie möglich optimieren können.

7.1 Methode

Um herauszufinden wo optimierungsbedarf besteht, haben wir uns darauf konzentriert, mit dem Geschäftsführer der Fahrschule zu sprechen. Jedoch haben wir uns auch vorher die bereits bestehende Website genau angeschaut, um zu begutachten, ob man dort etwas verändern könnte. Auch haben wir mit den Fahrschülern geredet und da ein paar aus unserem Team mal selbst bei dieser Fahrschule waren, konnten wir auch selbst einschätzen, wo es Probleme gibt.

7.2 Problembeschreibung

Bei unserer Analyse sind wir auf ein Problem gestoßen, dass sich auf beiden Seiten des Unternehmens bemerkbar macht. Fahrschüler und Fahrlehrer können nicht effizient die Termine für Fahrstunden ausmachen. Es stellt ein größeres Problem für die Fahrschüler dar, da manche von ihnen lange auf einen Termin für ihre nächste Fahrstunde warten müssen. Sie müssen zurzeit ihren Fahrlehrer persönlich ansprechen und aufsuchen, auch wenn sie es zeitlich nicht regelmäßig schaffen. Dieser ineffiziente Prozess hindert gegebenenfalls die Fahrschüler daran besser beim Fahren zu werden, da ihnen die Routine fehlt. Somit würden sie im Durchschnitt länger brauchen, um ihren Führerschein zu absolvieren.

7.3 Ansatz zur Lösung des Problems

Eine Lösung, die wir uns überlegt haben, wäre ein System, in dem sich Fahrlehrer und Fahrschüler anmelden können. Das System wäre auch auf der Webseite der Fahrschule zu finden, wofür Anmeldedaten benötigt werden. In diesem System geben Fahrlehrer Zeitblöcke frei, wie es ihnen am besten passt. Sie können auch Zeiten sperren, falls sie mal Urlaub haben. Die Fahrschüler können sich anmelden und die Zeiten, die noch frei sind buchen und auch den Abholort einsehen. Zusätzlich wäre es gut, dass der Fahrschüler sehen kann, wann die Theoriestunden sind, weil jeder Fahrschüler 14 Doppel-Stunden Theorie als Pflicht besuchen muss.

Diese Lösung würde die Planung der Fahrstunden maßgeblich vereinfachen und effizienter gestalten. So dauert es nämlich nur wenige Minuten einen neuen Termin zum Fahren auszumachen und der Fahrschüler braucht weniger Zeit für seinen Führerschein.

8 Zusammenfassung der Anforderungen und Zielgruppe

Erstellt von: Alina

Die Zielgruppe der Anwendung sind vor allem die Kunden, die potentiellen Kunden und die Fahrlehrer selbst. Das System ist vor allem für die vorhandenen Kunden wichtig und da es sich hier um Fahrschüler handelt, steht im Mittelpunkt, dass sie regelmäßig und leichter Termine für Fahrstunden mit ihrem Fahrlehrer buchen können. Die wichtigen Anforderungen von unserem Auftraggeber waren, dass es ein System ist, bei dem die Fahrlehrer bestimmte Zeitblöcke freigeben können, die die Fahrschüler dann buchen können. Für potentielle Kunden ist es nur relevant zu sehen, wo sich die Fahrschule befindet und wann die Theoriestunden und Bürozeiten sind.

Weitere Anforderungen waren, dass die Anwendung zu den Terminen für die Fahrstunden, nur in einem passwortgeschützten Bereich für die Fahrschüler und Fahrlehrer zugänglich sein sollen. Das Design der Anwendung, soll mit den bereits vorhandenen Webseiten der Fahrschule übereinstimmen, vor allem in Farbe, Aufbau und natürlich Sprache, damit alles einheitlich ist. Eine leichte Bedienung und Übersichtlichkeit der Architektur sind im Design von Relevanz. Es soll auch nicht zu kompliziert erscheinen, damit die Anwendung eine einfache Benutzung hergibt und möglichst keine Probleme auftreten. Falls doch Probleme auftreten sollten mit der Anwendung an sich, wird ein Kontakt bereitgestellt, für Hilfestellung und Problemlösung, mittels mailto.

Die Anwendung soll zudem ausführlich erklärt werden, damit die Fahrschüler und die Fahrlehrer das System optimal und effizient nutzen können.

9 Erläuterung der Methoden für die erhobenen Anforderungen

Erstellt von: Alina

9.1 Zweck der Software-Anforderungsspezifikation und Dokumentation

Die Software-Anforderungsspezifikation hat bei unserem Projekt den Zweck, dass wir einen guten Überblick über unsere Aufgabe behalten. Es ist wichtig erstmal zu wissen was genau von unserem Auftraggeber verlangt wird und dann an den Anforderungen festzumachen, was bearbeitet werden muss. Die Anforderungsspezifikation hilft uns, nicht zu sehr von dem abzuweichen, was unser Auftraggeber von uns verlangt, damit wir ihn zufrieden stellen können. Um herauszufinden, was genau in die Anforderungsspezifikation aufgenommen werden soll, haben wir uns mit dem Geschäftsführer mehrfach getroffen und die Punkte abgearbeitet. So können wir Missverständnisse vermeiden und die Anforderungen und die Implementierung nachverfolgen. Meistens standen uns gewisse Anforderungen offen, solange unsere Umsetzung zu den restlichen Anforderungen passt. Als Beispiel waren uns die Entwicklungstools von ihm nicht vorgegeben, weswegen wir diese aus unserem Modul entnommen haben.

9.2 Methoden zur Modellierung

Bei der Implementierung der Anforderungen hilft uns das Erstellen von UML-Diagrammen.

Zuallererst haben wir ein Anwendungsfalldiagramm zur Darstellung der Interaktionen zwischen den Akteuren und dem System erstellt. Es gibt einen Gast-User, der sich einloggen muss, um entweder die Rolle des Fahrschülers oder des Fahrlehrers einzunehmen. In dem Diagramm wird dargestelt, welcher Akteur was in dem System können soll. Hauptsächlich soll der Fahrschüler Fahrstunden buchen und sehen wie viele Stunden absolviert wurden. Der Fahrlehrer hingegen, soll neue Fahrschüler registrieren, sie löschen können, offene Fahrstunden eintragen und gebuchte Fahrstunden annehmen oder ablehnen. Beide Akteure sollten sich abmelden können, um wieder in der Gast-Rolle zu landen.

Auf das Anwendungsfalldiagramm folgen Aktivitätsdiagramme, die zeigen, wie der Fahrlehrer und wie der Fahrschüler mit dem System interagiert, um die möglichen Interaktionen funktionaler darzustellen. Bei beiden Akteuren wird erstmal, nachdem Aufrufen der Buchungsseite, geprüft, ob der Nutzer eingeloggt ist oder nicht. Danach wird die entsprechende Benutzeroberfläche und ihre Funktionen angezeigt. Die jeweiligen Änderungen, die durch einen Akteur vorgenommen wurden, werden in der Datenbank gespeichert.

Da die Aktivitätsdiagramme Klassen enthalten, konnten wir auch ein Klassendiagramm erstellen. Dort gibt es Unterschiede, welcher Akteur auf welche Klassen zugreifen kann, was wichtig zur Unterscheidung ist, wie es die Aktivitätsdiagramme zeigen. Der Lehrer kann auf die Klassen User, Wochenplan, Blocker, Termin, Schüler und Theoriestunde zugreifen. Der Schüler greift hingegen nur auf die Klassen User, Termin und Theoriestunde zu. User ist erforderlich für den Login auf beiden Seiten.

Die UML-Diagramme ergänzen sich gegenseitig, indem sie verschiedene Perspektiven des Systems darstellen, mit denen man effizienter arbeiten kann. So erhält man einen besseren Überblick über dem, was implementiert werden muss. Durch den Einsatz von UML-Diagrammen und die Erstellung einer umfassenden Software-Anforderungsspezifikation konnten die erhobenen Anforderungen effektiv modelliert und dokumentiert werden. Dies gewährleistet eine strukturierte und nachvollziehbare Entwicklung des Systems.

10 Erläuterung Prototyp zu UML

Erstellt von: Laura

10.1 Anwendungsfalldiagramm

Unsere Anwendung ist in mehrere Bereiche geteilt, die jeweils eine Gruppe von Anwendungsfällen abdecken. Aus Zeitgründen wurden in unserem Prototyp nicht alle dieser Fälle implementiert. Man vergleiche die Navigationsleiste der Webseite mit unserem Anwendungsfalldiagramm (siehe Spec Abschnitt 3.1).

- Startseite
  - Gast: Standorte auf einer Karte sehen
  - Gast: Informationen zu Theoriestunden sehen
- Personeneigenschaften
  - Fahrschüler: Sehen, wie viele Stunden schon absolviert wurden (nicht implementiert)
- Buchung
  - Fahrschüler: Fahrstunde buchen
  - Fahrlehrer: Gebuchte Fahrstunde annehmen oder ablehnen
- Kalender
  - Fahrschüler: Offene Fahrstunden seines Lehrers sehen (nicht implementiert)
  - Fahrlehrer: Offene Fahrstunden eintragen (nicht implementiert)
- Adminseite
  - Fahrlehrer: Fahrschüler löschen (nicht implementiert)
  - Fahrlehrer: Informationen zu Theoriestunden eintragen (nicht implementiert)
  - Fahrlehrer: Überblick seiner Schüler sehen (nicht implementiert)
- Login
  - Fahrschüler/Fahrlehrer: Sich einloggen
  - Fahrlehrer: Neuen Fahrschüler registrieren
- Logout (ersetzt Login in der Navigationsleiste, wenn Nutzer eingeloggt ist)
  - Fahrschüler/Fahrlehrer: Sich abmelden

Darüber hinaus leitet „FFM Hauptseite“ auf die bestehende Webseite der FFM.

10.2 Aktivitätsdiagramme

Unsere Aktivitätsdiagramme (siehe Spec Abschnitt 3.2) definieren den Ablauf der Anwendungsfälle „Fahrschüler: Fahrstunde buchen“ und „Fahrlehrer: Offene Fahrstunden eintragen“. Dieser ist in beiden Fällen ähnlich. Wenn der Nutzer die jeweilige Seite aufruft, prüft das System zunächst, ob er eingeloggt ist. Wenn nicht, erfolgt eine Weiterleitung auf die Login-Seite. Ist er eingeloggt, wird die Benutzeroberfläche angezeigt, welche genau die Formularfelder zur Verfügung stellt, die der Nutzer für die im Diagramm definierten Eingaben benötigt. Nach dem Bestätigen des Formulars wird dann ein Objekt erstellt und in der Datenbank gespeichert, und eine Bestätigung wird angezeigt.

10.3 Klassendiagramm

Die im Klassendiagramm (siehe Spec Anhang) definierten Klassen sind die Datengrundlage der Anwendung. So bilden sich daraus alle Tabellen der Datenbank. Das Frontend der Anwendung agiert unter der Annahme, dass die Daten in diesem Format aus der Datenbank gelesen und in sie geschrieben werden (Schnittstellendefinition). So wird zum Beispiel beim Einloggen-Fall geprüft, ob ein Eintrag in der User-Tabelle existiert, der mit den vom Nutzer eingegeben Namen und Passwort übereinstimmt, und beim Registrieren wird ein entsprechender Eintrag im selben Format erstellt.

Da das Klassendiagramm fachlich ist, weicht die technische Implementierung der Klassen etwas ab. Im Diagramm sind Schüler und Lehrer separate Klassen, die von der Klasse User erben. Vererbung ist in einer relationalen Datenbank ohne weiteres nicht möglich. Daher hatten wir die Wahl, entweder zwei User-Tabellen für die beiden Rollen zu erstellen, oder eine Tabelle User mit einem Boolean, um die Rolle des Nutzers festzuhalten. Letzteres erwies sich als vorteilhafter, da somit auch einfacher sichergestellt werden konnte, dass keine doppelten Nutzernamen im System vorhanden sind.

Es wurden hier ebenfalls aus Zeitgründen nicht alle Klassen implementiert. Nur die User-Tabelle und die Termin-Tabelle ist vorhanden, und zwischen diesen beiden bestehen in unserer Datenbank keine Beziehungen, da die entsprechenden Funktionen wie das Zuweisen von Lehrer und Schüler auch im Frontend nicht implementiert sind. Die Implementation weiterer Klassen würde der bisherigen Implementation allerdings sehr ähnlich sein.

10.4 Zustandsdiagramm

Das Zustandsdiagramm (siehe Spec Anhang) definiert, in welchen Fällen der Status einer Terminbuchung wechselt und welche Aktionen in den jeweiligen Zuständen durch Fahrlehrer getätigt werden können. Dieses Diagramm verglichen wir später mit dem Prototyp, um sicherzustellen, dass das Verhalten korrekt ist.

11 Erklärung wichtiger Code Ausschnitte

Erstellt von: Fabian

11.1 Leaflet einbindung

```
var map1 = L.map('map1').setView([53.43522, 8.82122], 15);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a
    ↪ href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
    ↪ contributors'
}).addTo(map1);

L.marker([53.43522, 8.82122]).addTo(map1)
    .bindPopup('Poststraße 13, 27616 Beverstedt')
    .openPopup();

var map2 = L.map('map2').setView([53.5033, 8.5996], 15);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a
    ↪ href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
    ↪ contributors'
}).addTo(map2);

L.marker([53.5033, 8.5996]).addTo(map2)
    .bindPopup('Weserstraße 46, 27572 Bremerhaven')
    .openPopup();
```

Dieser JavaScript-Code verwendet die Leaflet-Bibliothek, um zwei interaktive Karten mit OpenStreetMap-Tiles zu erstellen und darauf Marker zu platzieren.

```
var map1 = L.map('map1').setView([53.43522, 8.82122], 15);
```

Hier wird eine Karte in dem HTML-Element mit der ID map1 erstellt und dann wird die Ansicht (Kartenmittelpunkt und Zoomstufe) auf die angegebenen Koordinaten (Breitengrad 53.43522, Längengrad 8.82122) gesetzt mit der Zoomstufe 15.

```
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
```

Dieser Aufruf in der Leaflet-Bibliothek erstellt eine neue Tile-Layer, die Kacheln von OpenStreetMap verwendet. Die URL-Vorlage gibt das Muster an, nach dem die Kachel-URLs generiert werden. Die Platzhalter s, z, x und y werden durch die entsprechenden Werte für Subdomäne s, der Zoomstufe z, die X x- und Y y-Koordinaten der Kachel ersetzt.

```
attribution: '&copy; <a
↪  href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
↪  contributors'
```

attribution fügt eine Quellenangabe hinzu, in diesem Fall OpenStreetMap.

```
}).addTo(map1);
```

Hier wird der Markter zu der Karte ”map1”hinzugefügt.

```
.bindPopup('Poststraße 13, 27616 Beverstedt')
.openPopup();
```

Damit wird ein Popup mit dem angegebenen Text an den Marker gebunden. Und mittels ”.openPopup”wird das Popup sofort angezeigt sobald die Karte geladen ist.

```
var map2 = L.map('map2').setView([53.5033, 8.5996], 15);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a
    ↪  href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
    ↪  contributors'
}).addTo(map2);

L.marker([53.5033, 8.5996]).addTo(map2)
    .bindPopup('Weserstraße 46, 27572 Bremerhaven')
    .openPopup();
```

Hier ist der gleiche Code, aber angepasst mit anderen Daten für die ”map2Karte.

11.2 buchung.js

Hier gibt es noch eine kurze Erklärung und codeteile zu unserer buchung.js.

```
if (localStorage.getItem('loggedInAs') === "Lehrer") {
```

Hier wird überprüft, ob der Benutzer als 'Lehrer' eingeloggt ist.

```
document.getElementById("container-schueler").style.display = "none";
document.getElementById("container-lehrer").style.display = "block";
```

Sollte das der fall sein dann wird die Schüler-Oberfläche ausgeblendet und die Lehrer-Oberfläche angezeigt.

```
const xmlHttpRequest = new XMLHttpRequest();
xmlHttpRequest.open("GET",
↪  "https://informatik.hs-bremerhaven.de/docker-swe2-2024-team05-web
    /cgi-bin/get-vorgeschlagene-termine.sh");
xmlHttpRequest.onreadystatechange = () => {
    if (xmlHttpRequest.readyState === 4 && xmlHttpRequest.status ===
    ↪  200) {
```

Wenn der Schüler einen Termin vorschlägt, dann wird hier die Anfrage gesendet, die der Schüler vorgeschlagen hat.

```
const terminListe = [];
const termine = xmlHttpRequest.responseText.split("\n");
termine.forEach(termin => {
    if (termin === '' || termin.startsWith("id")) {
        return;
    }

    const id = termin.split("\t")[0];
    const datum = termin.split("\t")[1];
    const zeitVon = termin.split("\t")[2];
    const zeitBis = termin.split("\t")[3];
    const abholort = termin.split("\t")[4];

    terminListe.push(`
        <h2>${datum} ${zeitVon} - ${zeitBis}</h2>
        <label for="abholort-${id}">Abholort:</label>
        <input type="text" id="abholort-${id}" class="form-style"
        ↪  value="${abholort}">
        <a href="#" class="btn"
        ↪  onclick="bestaetigen(${id})">Bestätigen</a>
        <a href="#" class="btn" onclick="ablehnen(${id})">Ablehnen</a>
    `);
});
```

In diesem Abschnitt werden die Antworten vom Server verarbeitet, um die Termindaten zu extrahieren und in eine Liste zu formatieren.

```
document.getElementById("container-lehrer").innerHTML = `
    <div class="box">
        ${terminListe.join("<br>")}
    </div>
`;
}
};
xmlHttpRequest.send();
```

Hier wird die formatierte Liste von Terminen in die Lehrer-Oberfläche eingefügt.

```
} else {
    document.getElementById("submit").addEventListener("click", () => {
        const datum = document.getElementById("datum").value;
        const zeitVon = document.getElementById("zeit-von").value;
        const zeitBis = document.getElementById("zeit-bis").value;
        const abholort = document.getElementById("abholort").value;

        const url =
        ↪ `https://informatik.hs-bremerhaven.de/docker-swe2-2024-
            team05-web/cgi-bin/create-termin.sh?datum=${datum}&
            zeitvon=${zeitVon}&zeitbis=${zeitBis}&abholort=${abholort}`;

        const xmlHttpRequest = new XMLHttpRequest();
        xmlHttpRequest.open("GET", url);
        xmlHttpRequest.onreadystatechange = () => {
            if (xmlHttpRequest.readyState === 4 && xmlHttpRequest.status
            ↪ === 200) {
                document.getElementById("container-schueler").innerHTML
                ↪ = `
                    <div class="box">
                        Fahrstunde wurde erfolgreich gebucht!<br>
                        Datum: ${datum}<br>
                        Beginn: ${zeitVon}<br>
                        Ende: ${zeitBis}<br>
                        Abholort: ${abholort}<br>
                        Status: Vorschlag<br>
                        Bitte warten Sie auf eine Bestätigung des
                        ↪ Fahrlehrers.
                    </div>
                `;
            }
        };
        xmlHttpRequest.send();
    });
}
});
```

Falls der Benutzer ein Schüler ist, wird ein Event Listener hinzugefügt, der beim Klick auf den 'Submit'-Button eine Anfrage sendet, um einen neuen Termin vorzuschlagen.

```
function bestaetigen(id) {
    const abholort = document.getElementById(`abholort-${id}`).value;
    const xmlHttpRequest = new XMLHttpRequest();
    xmlHttpRequest.open("GET",
    ↪ `https://informatik.hs-bremerhaven.de/docker-swe2-2024-
        team05-web/cgi-bin/termin-bestaetigen.sh?id=
        ${id}&abholort=${abholort}`);
    xmlHttpRequest.onreadystatechange = () => {
        if (xmlHttpRequest.readyState === 4 && xmlHttpRequest.status ===
        ↪ 200) {
            window.location.replace(".");
        }
    };
    xmlHttpRequest.send();
}

function ablehnen(id) {
    const xmlHttpRequest = new XMLHttpRequest();
    xmlHttpRequest.open("GET",
    ↪ `https://informatik.hs-bremerhaven.de/docker-swe2-2024-
        team05-web/cgi-bin/termin-ablehnen.sh?id=${id}`);
    xmlHttpRequest.onreadystatechange = () => {
        if (xmlHttpRequest.readyState === 4 && xmlHttpRequest.status ===
        ↪ 200) {
            window.location.replace(".");
        }
    };
    xmlHttpRequest.send();
}
```

Dieser Codeabschnitt sorgt für die Bereitstellung von Funktionen für Lehrer, um Termine zu bestätigen oder abzulehnen.

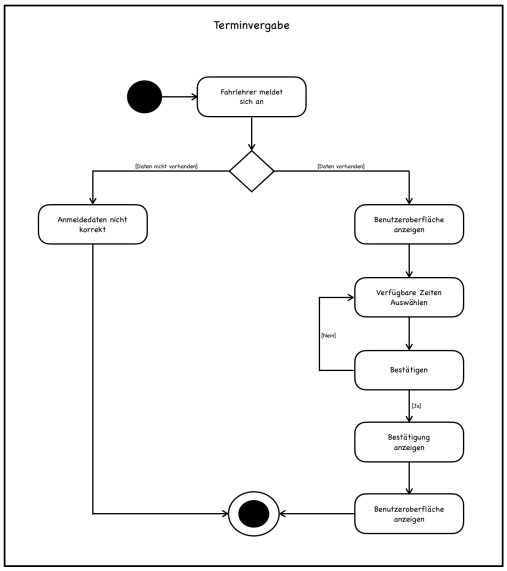
12 Qualitätssicherung

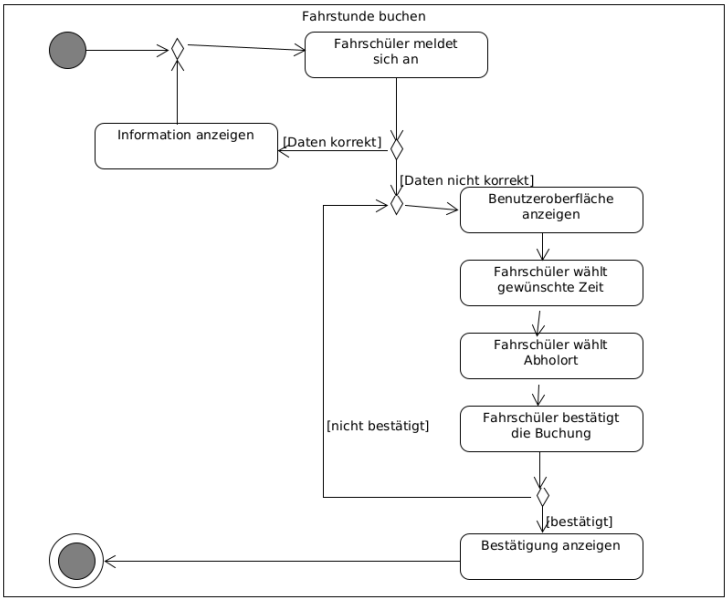
Erstellt von: Laura

12.1 Qualität der UML-Modellierung

Wir führten für unsere Anforderungsmodellierung eine Qualitätssicherung durch, in dem wir die erstellten UML-Diagramme überprüften.

Dazu verwendeten wir die Checkliste, die uns im Rahmen des Moduls zur Verfügung gestellt wurde. Insbesondere die Aktivitätsdiagramme wurden darauf noch einmal überarbeitet. Schwimmbahnen und Objekte fehlten, obwohl sie gefordert waren, und an manchen Stellen waren syntaktische Fehler. Im Folgenden die erste Version der Diagramme:





Die zweite, überarbeitete Version der Diagramme (siehe Spec Abschnitt 3.2) zeigen nun die Aktionen von Nutzer und System deutlicher durch die Trennung mit Schwimmbahnen. Außerdem werden die dabei erstellten Objekte ebenfalls dargestellt. Allgemein stellen die Diagramme ihre Anwendungsfälle nach dem Qualitätssicherungsprozess nun präziser dar.

12.2 Qualität der Anwendung

Für unseren Prototypen führten wir ebenfalls eine Qualitätssicherung durch. Dazu definierten wir Testfälle für den Anwendungsfall „Sich einloggen“.

ID	Beschreibung	Vorbedingungen	Schritte	Daten	Erwartetes Ergebnis
01	Login Schüler	Startseite ist aufgerufen	1. In der Navigation „Login“ auswählen 2. Daten eingeben 3. „Login“-Button klicken	Username: test-schueler Passwort: demo	Nutzer wird als Schüler eingeloggt und auf die Startseite zurückgeleitet
02	Login Lehrer	Startseite ist aufgerufen	1. In der Navigation „Login“ auswählen 2. Daten eingeben 3. „Login“-Button klicken	Username: test-lehrer Passwort: demo	Nutzer wird als Lehrer eingeloggt und auf die Startseite zurückgeleitet
03	Login Invalide Daten	Startseite ist aufgerufen	1. In der Navigation „Login“ auswählen 2. Daten eingeben 3. „Login“-Button klicken	Username: gibt-es-nicht Passwort: gibt-es-nicht	Eine Fehlermeldung wird angezeigt
04	Login Leeres Formular	Startseite ist aufgerufen	1. In der Navigation „Login“ auswählen 2. „Login“-Button klicken		Eine Fehlermeldung wird angezeigt

Wir führten darauf die Testfälle am Prototypen durch, sobald die Login-Funktion implementiert war. Unsere Tests lieferten die erwarteten Ergebnisse und zeigten so, dass die Login-Funktion funktioniert. Nach Abschluss des Prototypen führten wir diese Tests ein weiteres Mal aus, um sicherzustellen, dass Änderungen an anderen Komponenten keinen Einfluss auf diese Anwendungsfälle genommen hat.

Über diese konkreten Testfälle hinaus prüften wir, ob das Verhalten des Prototypen unseren Erwartungen entspricht, indem wir die Anwendungsfälle durchspielten und dabei sowohl auf funktionale als auch nicht-funktionale Anforderungen (zum Beispiel Geschwindigkeit) achteten.

13 Reflexion und Fazit

Erstellt von: Laura

In diesem semesterlangen Projekt haben wir viel über die Planung, Modellierung und Durchführung eines Software-Entwicklungsprojekts gelernt.

Die Modellierung mit UML-Diagrammen hat uns im Projekt sehr geholfen. Die Diagramme konnten als Ergebnis der Diskussionen um die Anforderungen festgehalten werden, so dass alle Beteiligten wirklich die selben Erwartungen an die Anwendung haben. Insbesondere das Anwendungsfalldiagramm diente auch laufend als Erinnerung für die besprochenen funktionalen Anforderungen, die sonst im Laufe des Semesters vergessen werden würden. Die Aktivitätsdiagramme halfen ebenfalls bei der genauen Definition von Prozessabläufen, wobei diese eine ziemlich komplexe Syntax aufweisen, die in unserem spezifischen Fall auch nicht unbedingt zur besseren Übersicht beitrug. Bei längeren und komplizierteren Anwendungsfällen könnten diese Syntaxelemente sich jedoch als nützlich erweisen. Über unsere wissenschaftliche Recherche erhielten wir weitere Einblicke in die Vor- und Nachteile der UML, welche unsere Erfahrungen teilweise bestätigten.

Da das Projekt in der Wirklichkeit keine Anwendung finden wird, war der Kundenkontakt abgesehen von einem initialen Treffen sehr eingeschränkt. Daher wurden feinere Anforderungen teilweise von uns selbst definiert. Uns ist bewusst, dass das in der Realität natürlich mit dem Kunden abgesprochen werden würde, und diese feineren Anforderungen dann auch schriftlich oder graphisch festgehalten werden sollten. Die Methoden dafür haben wir in diesem Projekt dennoch erlernt und könnten sie in realen Projekten auch dafür einsetzen.

Das Tool OpenProject haben wir für unser Projekt nur mäßig genutzt. Mit häufigen Treffen, in denen klare Aufgaben für den Zeitraum bis zum nächsten Treffen definiert, die Aufgaben des vorigen Zeitraums besprochen, und an anstehende Fristen erinnert wurde, hatten wir bereits einen sehr guten Überblick über den Projektfortschritt. OpenProject schien als Zusatz dazu nicht allzu hilfreich und war aufgrund seiner Komplexität auch mit höherem Zeitaufwand verbunden. Bei größeren, komplizierteren Projekten unter anderem im Unternehmenskontext erscheint uns ein solches Tool eher als sinnvoll.

Zur gemeinsamen Entwicklung des Prototypen und Zusammentragen der Dokumente verwendeten wir Git. Dadurch ließ sich Arbeit gut aufteilen und Änderungen nachverfolgen.

Ein Problem in unserem Projekt war letztendlich eine Unterschätzung des Arbeitsaufwands für die Umsetzung unserer Anforderungen. Besonders die Erfassung und Berechnung von freien Zeiten der Fahrlehrer zeigte im Entwicklungsprozess unerwartete Komplexität auf, da Wochenpläne, Blockzeiten und gebuchte Termine berücksichtigt werden mussten. Auch das Aufsetzen des Grundgerüsts der Seite mit Datenbankanbindung nahm einige Zeit in Anspruch. Als Folge daraus benötigte die Entwicklung des Prototypen mehr Zeit als erwartet und lief bis kurz vor dem Abgabetermin weiter. Dementsprechend blieb auch nicht viel Zeit für umfangreiche Qualitätssicherung oder ein Review mit dem Kunden. Durch weitere Erfahrung in der Softwareentwicklung sollten wir in Zukunft ein bessers Gefühl für den Zeitaufwand einer Aufgabe erhalten.

Insgesamt sind wir mit unserer Projektarbeit zufrieden, auch wenn es ein paar Hindernisse gab und wir einige Aspekte in zukünftigen Projekten anders gestalten würden. Wir konnten den Nutzen sowie den Aufwand von Tools wie OpenProject, UML-Diagrammen, Interviews zur Anforderungserhebung, Git zur Entwicklung und Testfälle sehen und können somit in Zukunft fundierte Entscheidungen darüber treffen, welche Tools wir einsetzen wollen und wie wir damit umgehen.

14 Wissenschaftlicher Artikel

Erstellt von: Alle

Für unsere wissenschaftliche Recherche verwendeten wir vier Artikel (Beyer und Hesse 2002, Randak 2011, Götze und Kattanek 2001, citeartikel4) zu dem Thema Vor- und Nachteile der UML-Modellierung.

Erstellt von: Laura Halbeck:

Die UML bietet eine standardisierte und gut lesbare Möglichkeit zur Modellierung, die es allen Beteiligten eines Projekts ermöglicht, die Diagramme zu verstehen, sofern sie sich bereits mit UML beschäftigt haben. Diese einheitliche Sprache erleichtert die Kommunikation im Team und schafft Klarheit. Auch Personen ohne Vorkenntnisse können einen groben Überblick gewinnen, auch wenn sie möglicherweise Schwierigkeiten haben, die Unterschiede zwischen spezifischen Syntaxelementen vollständig zu erfassen.

Ein weiterer Vorteil der UML ist ihre generische Natur. Sie kann für verschiedene Anwendungsfälle verwendet werden, muss aber dadurch auch manchmal erweitert oder modifiziert werden, um den speziellen Anforderungen gerecht zu werden. Klassendiagramme sind besonders nützlich und in vielen Fällen gut geeignet, da sie die Struktur von Systemen und deren Beziehungen klar darstellen. Im Gegensatz dazu können Aktivitätsdiagramme nicht alle Prozesse optimal darstellen. Sie sind nützlich für die Modellierung von Arbeitsabläufen, stoßen jedoch bei komplexeren Abläufen an ihre Grenzen.

Die UML bietet eine flexible und weit verbreitete Methode zur Visualisierung und Kommunikation von Systemstrukturen und -prozessen, die jedoch je nach Anwendung angepasst werden muss. Somit ist sie ein sehr hilfreiches Werkzeug in der Softwareentwicklung und im Projektmanagement, das trotz kleinerer Einschränkungen großen Nutzen bietet.

Erstellt von: Fabian Tober:

Alle Artikel betonen die Vielseitigkeit und Anpassungsfähigkeit der Unified Modeling Language, die durch verschiedene Diagrammtypen ermöglicht wird. Die Verwendung der Unified Modeling Language ist besonders vorteilhaft für die Modellierung komplexer Systeme wie eines Fahrschulmanagementsystems, das sowohl statische Elemente (z. B. Datenbankstrukturen für die Verwaltung von Schülern und Fahrlehrern) als auch dynamische Prozesse (z. B. Terminplanung und Buchungsabläufe) umfasst. Durch die Modularisierung und Wiederverwendbarkeit der Unified Modeling Language kann die Handhabbarkeit und Wartbarkeit des Systems verbessert und die Kommunikation im Entwicklungsteam erleichtert werden.

Jedoch gibt es auch Nachteile. Die allgemeine Natur von Unified Modeling Language kann dazu führen, dass sie für spezifische Aufgaben nicht ausreichend präzise ist. Für ein Fahrschulmanagementsystem könnten spezielle Anpassungen erforderlich sein, um die spezifischen Prozesse und Anforderungen angemessen darzustellen. Zudem kann die Komplexität der Unified Modeling Language-Profile und die Notwendigkeit tiefgehenden technischen Wissens den Entwicklungsprozess erschweren und die Wartung komplizieren. Besonders die dynamische Modellierung, die für die Prozessabläufe im Fahrschulmanagementsystem von Bedeutung ist, befindet sich noch im Experimentierstadium und kann Herausforderungen in Bezug auf Konsistenz und Übersichtlichkeit mit sich bringen.
Insgesamt bietet die Unified Modeling Language wertvolle Werkzeuge für die Modellierung eines Fahrschulmanagementsystems, insbesondere durch ihre Vielseitigkeit und Modularisierung. Entwickler sollten jedoch die generischen Einschränkungen und die Komplexität der Unified Modeling Language-Profile berücksichtigen und gegebenenfalls spezialisierte Anpassungen vornehmen, um die spezifischen Anforderungen des Projekts zu erfüllen.

Erstellt von: Alina Hoeborn:

Die UML-Modellierung bietet viele Vorteile, die sie zu einem bedeutenden Werkzeug in der Softwareentwicklung machen. Die standardisierte Notation von UML ernöglicht eine einheitliche Kommunikation zwischen verschiedenen Teams und Stakeholdern. Diese Standardisierung fördert das gemeinsame Verständnis und erleichtert die Zusammenarbeit erheblich. Dadurch werden auch Missverständnisse minimiert und Projekte effizienter.
Die detaillierte und umfassende Dokumentation von Systemen, ist besonders wichtig für die Wartung und Weiterentwicklung, da auch komplexe Systeme klar und verständlich dargestellt werden können. Eine präzise Dokumentation verbessert die Nachverfolgbarkeit und Analyse von Systemen und trägt wesentlich zur Qualitätssicherung bei. UML kann in verschiedenen Phasen der Softwareentwicklung eingesetzt werden, von der Anforderungsanalyse bis zur Implementierung. Diese Flexibilität zeigt sich auch darin, dass UML in verschiedenen Branchen genutzt werden kann.
Es gibt jedoch einige Nachteile. Ein wesentlicher Kritikpunkt ist die Komplexität der vollständigen Nutzung aller UML-Funktionalitäten. Dies kann überwältigend sein und stellt für Anfänger eine Hürde dar. UML erfordert viel Aufwand, was die Effizienz und Akzeptanz beeinträchtigt. In der Anwendung wird oft nur eine begrenzte Auswahl der UML-Konstrukte genutzt, was zu Komplexität führt. Die damit verbundene steile Lernkurve könnte dazu führen, dass die Effizienz in der Praxis benachteiligt wird.
Trotz dieser Nachteile überwiegen die Vorteile der UML-Modellierung, insbesondere wenn man bereit ist, sich gründlich in die Materie einzuarbeiten. Die langfristigen Vorteile machen UML zu einem wertvollen Werkzeug in der Softwareentwicklung.

Literatur

Beyer, Mario und Wolfgang Hesse (2002). „Einsatz von UML zur Software-Prozeßmodellierung“. In: *GI-Softwaretechnik-Trends* 22, S. 4–11. URL: https://www.mathematik.uni-marburg.de/~hesse/papers/B_H_02.pdf.

Götze, Marco und Wolfram Kattanek (2001). „Erfahrungen mit der UML beim Entwurf von Kfz-Steuerungen.“ In: S. 87–98.

Randak, Andrea (2011). „ATL4pros: introducing native UML profile support into the ATLAS transformation language“. In.

15 Anhang

- Teamregeln
- Kooperationsvereinbarung
- SPEC
- Arbeitspakete
- Autorenverzeichnis
- Webseiten Login Standard Benutzer