IU Internationale Hochschule
Weiterbildung: “Software Engineering - Python”
Modul : DLMCSPSE01_D - Projekt: Software Engineering
Tutor: Prof. Dr. David Kuhlen

Prüfungsleistung: Portfolio

Abstract

Eingereicht am 16.09.2025.

Verfasser:
Djahan Bayrami Latran
Denninger Straße 198
81927 München
E-Mail: djahan.latran@gmail.com
Matrikelnummer: UPS10672478

Im Rahmen des Projekts “Software Engineering” wurde eine Anwendung zur Visualisierung von unterschiedlichen Algorithmen erstellt. Hierbei wurde die Programmiersprache “Python” genutzt, da diese eine umfangreiche Auswahl an passenden Bibliotheken besitzt und sich durch ihre simple Syntax und automatisiertes Speichermanagement eignet, um frühe Prototypen zu erstellen. Um die Algorithmen zu visualisieren, wurde die Bibliothek “Pygame” genutzt, mit der interaktive 2D-Oberflächen erstellt werden können.
Das Ziel der Anwendung war es, die Konzepte und Funktionsweisen von diversen Such-, Sortier- und Graphenalgorithmen durch geeignete visuelle Aufbereitung besser nachvollziehbar zu machen. Dabei lag der Fokus auf einer intuitiven Benutzeroberfläche und einer interaktiven Steuerung der Algorithmen. Die Ziel Nutzergruppe waren Lernende der Informatik, Softwareentwicklung oder ähnlichen Bereichen.
Die Umsetzung wurde mit einer Definition des Funktionsumfangs initiiert. Vorgesehen waren die Implementierung von verschiedenen Algorithmen, die Funktion, diverse Parameter, wie z.B. die Ablaufgeschwindigkeit ändern zu können, und die Anzeige von zusätzlichen Informationen des gewählten Algorithmus zu ermöglichen. Zur Umsetzung wurde ein Zeitplan mit verschiedenen Phasen und Meilensteinen erstellt. Gleichzeitig wurde ein Testkonzept entwickelt, das sowohl automatisierte Unit-Tests als auch manuelle Tests vorsah.
Nach der Anforderungserhebung, der Festlegung zentraler Geschäftsregeln und der Umsetzung eines Risikomanagements ging es über die Festlegung der Anwendungsarchitektur. Es wurden diverse UML Diagramme erstellt, die das Klassensystem und zentrale Prozesse der Anwendung besser veranschaulichen. Als erstes wurden einfache Prototypen der Kernfunktionalitäten entwickelt. Zunächst wurde das Animationsfenster prototypisch entwickelt, um die Darstellung der Algorithmus-Abläufe frühzeitig zu testen. Als nächstes wurde ein Prototyp des User Interfaces erstellt, wo der nächste Meilenstein darin lag, das Animationsfenster in dieses funktionsfähig zu integrieren. Anschließend wurden das Auswahlmenü, die Steuerungslogik mit Interaktionen wie Starten und Pausieren und die Infobereiche implementiert. So wurde die Kernfunktionalität der Anwendung Stück für Stück erreicht. Am Ende folgte ein umfangreiches Refactoring. Nebenbei wurde regelmäßig auf das Feedback vom Tutor eingegangen und entsprechende Anpassungsmaßnahmen ergriffen. In der letzten Phase wurde die hybride Teststrategie durchgeführt und die Ergebnisse protokolliert.
Während der Anwendungsentwicklung ergaben sich diverse Herausforderungen. Z.B. erwies sich die Implementierung einer Geschwindigkeitssteuerung für die Algorithmen komplizierter als ursprünglich gedacht. Die Idee war, die FPS zu reduzieren, damit die einzelnen Schritte langsamer ablaufen. Dies hatte jedoch zur Folge, dass Reaktionen des gesamten UI’s verzögert wurden. Aus diesem Grund mussten der Ablauf der Algorithmus-Animation und

das Rendern der Benutzeroberfläche entkoppelt werden, um eine flüssige Bedienung auch während der reduzierten Algorithmus-Geschwindigkeit zu gewährleisten.
Die Integration des Animationsfensters in das Gesamtsystem war ein weiterer Meilenstein in der Entwicklung. Die 2D-Animation wurde mit Pygame umgesetzt. Eine passende Bibliothek für UI-Komponenten zu finden, die die Integration eines Pygame-Fensters ermöglicht, erwies sich als problematisch. Es stellte sich heraus, dass mit gängigen UI-Bibliotheken wie Tkinter oder PyQt dies nur schwer oder gar nicht umzusetzen sei. Aus diesem Grund fiel die Wahl auf pygame-gui. Die Bibliothek ist zwar vergleichsweise in ihrem Funktionsumfang begrenzt, jedoch bot sie eine pragmatische Lösung, da sie auf die Kompatibilität mit Pygame ausgelegt ist. Statt einem klassischen Architekturkonzept zu folgen, entsprach die Architektur der Anwendung anfangs lediglich dem Single-Responsibility-Prinzip (SRP). In einem der Tutorien wurde vom Tutor auf dieses Defizit hingewiesen. Daraufhin wurde der gesamte Entwurf der Anwendung nach dem MVC-Prinzip umstrukturiert. Dies stellte sich herausfordernder dar, als ursprünglich vermutet. Die strikte Entkopplung der Schichten konnte nicht ganz zufriedenstellend umgesetzt werden. Im Nachhinein wäre ein Ereignis-System hilfreich gewesen, auf das der Controller reagieren könnte (Observer Pattern). In der aktuellen Implementierung kennt die View den Controller und ruft mit der Controller-Instanz Methoden des Controllers auf, wenn bestimmte Kriterien erfüllt wurden. Hier hätte es noch Potenzial gegeben, die direkten Abhängigkeiten weiter zu entkoppeln. Auch wäre es in diesem Fall sinnvoller gewesen, so früh wie möglich einen Gesamt-Prototypen zu erstellen, um rechtzeitig zu erkennen, wo Abhängigkeiten entstehen könnten. Eine weitere Herausforderung war die Einhaltung des Zeitplans. Da das Modul im Rahmen einer freiwilligen berufsbegleitenden Weiterbildung bearbeitet wurde und die beruflichen Verpflichtungen flexible Zeiteinteilung erfordern, musste der ursprünglich festgelegte Zeitplan leider mehrmals geändert werden. Auch in der Finalisierungsphase kam es leider zu einer weiteren Verzögerung. Da es sich bei diesem Projekt um das erste umfangreiche Softwareprojekt handelte, war es schwierig den zeitlichen Aufwand der einzelnen Arbeitspakete richtig zu schätzen. Durch ein Zeit-Tracking der Arbeitspakete können zukünftige Zeitpläne hoffentlich realistischer erstellt werden - auch wenn berufliche Mehrarbeit nicht weit vorhersehbar ist.
Die Zielsetzung des Projekts war von Anfang an sehr klar. Das Ergebnis entspricht sowohl visuell als auch funktional dem Konzept und Anforderungen aus der frühen Phase. Trotz technischer Hürden bei der Integration des Animationsfensters in das GUI konnte das Ziel einer intuitiven Benutzeroberfläche erreicht werden. Durch regelmäßiges Einholen von Feedback des Tutors und Einarbeitung dessen, entstand ein großer Lernfortschritt - besonders in den Bereichen Softwarearchitektur und Design-Patterns. Hier wurde auch deutlich, dass frühzeitige, saubere Modularisierung spätere zeitintensive Anpassungen

vorbeugt. In den frühen Phasen der Prototypisierung wurde darauf leider nicht so streng geachtet. Bei zukünftigen Projekten wird dementsprechend anders gehandelt.
Zusammenfassend wurde das anfangs angestrebte Ziel einer funktionierenden Algorithmus-Visualisierungs-Applikation erreicht. Außerdem wurden wichtige Erkenntnisse in den Bereichen Softwarearchitektur und Design Patterns gemacht. Die Umstrukturierung von einer rein SRP-orientierten Architektur zu einer MVC-Schichtenarchitektur machte die Anwendung wartbarer und verständlicher. Es wurde gleichzeitig deutlich, dass die Umsetzung einer Entkopplung und Modularisierung anspruchsvoll ist und zusätzliche Strukturen wie Events benötigt werden, um konsequent auf alle Bereiche anwendbar zu sein. Das gesamte Projekt war demnach nicht nur eine technische Umsetzung, sondern eine ausgeprägte Lernerfahrung und wichtige Basis für zukünftige Projekte.