Lehrprojekt-Reflexionsberichte im Rahmen des Zertifikatsprogramms "Zukunftsorientierte Hochschullehre durch Technologieintegration" des Zentrums für Hochschullehre der Universität Bayreuth

Ausgabe VII – Mai 2026

Zertifikatsteilnehmer:
Dr. Nico Höllerich, Lehrstuhl für Robotik und Eingebettete Systeme

Fokus des Lehrprojekts:
KI als Programmier-Tutor

UNIVERSITÄT BAYREUTH

ZENTRUM FÜR HOCHSCHULLEHRE

Inhaltsverzeichnis

Zentrum für Hochschullehre

Die Lehrprojekt-Reflexionsberichte von Teilnehmenden des Zertifikatsprogramms "Zukunftsorientierte Hochschullehre durch Technologieintegration" werden vom Zentrum für Hochschullehre an der Universität Bayreuth veröffentlicht.

Das Zentrum für Hochschullehre bietet hochschuldidaktische Serviceleistungen für alle Lehrenden an. Die Aktivitäten und Angebote umfassen neben den klassischen Fortbildungsangeboten, Projekte zur Weiterentwicklung von Lehre mit Lehrstühlen, Lehrberatungen und Coachings von Einrichtungen und Einzelpersonen sowie Austauschformate im weiten Feld der Hochschullehre. Durch den Erwerb von Zertifikaten können alle Lehrenden ihr Engagement und ihre Kompetenzen im Bereich der Lehre sichtbarer machen. Somit versteht sich das ZHL als Servicestelle, die sich um jegliche Lehrthemen an der Universität Bayreuth kümmert. Alle Aktivitäten drehen sich ums Entwickeln, Unterstützen und Austauschen von Lehre nach den Wünschen und Anforderungen der Lehrenden.

Zertifikatsprogramm „Zukunftsorientierte Hochschullehre durch Technologieintegration“

Das Zertifikatsprogramm "Zukunftsorientierte Hochschullehre durch Technologieintegration" der Universität Bayreuth bietet Lehrenden eine strukturierte und individuelle Weiterbildungsmöglichkeit. Ziel des Programms ist es, die Integration moderner Technologien in der Hochschullehre zu fördern, um die Lehrqualität zu verbessern und den Lernprozess der Studierenden zu unterstützen.

Programmstruktur

Das Programm besteht aus folgenden Komponenten:

1. Individuelles Lernportfolio: Ein digitales Portfolio begleitet die Teilnehmenden durch das gesamte Programm. Es dient dazu, den individuellen Lernfortschritt zu dokumentieren und zu reflektieren. Das Portfolio bietet Raum für die schriftliche Bearbeitung relevanter Inhalte und unterstützt die Teilnehmenden dabei, ihre eigenen Entwicklungsziele zu verfolgen und zu evaluieren.
2. Beratungsgespräche: Individuelle Beratungsgespräche bieten den Teilnehmenden die Möglichkeit, maßgeschneiderte Unterstützung und Feedback zu erhalten. Diese Gespräche finden mit einer Hochschuldidaktikerin bzw. einem Hochschuldidaktiker statt und zielen darauf ab, die eigene Lehre weiterzuentwickeln. Zusätzlich werden Austauschphasen mit Kolleginnen und Kollegen organisiert, die Rückmeldungen zu den eigenen Ideen und Überlegungen ermöglichen. Durch Hospitationen erhalten die Teilnehmenden hochschuldidaktisches Feedback und können dadurch wichtige Erkenntnisse für die künftige Lehre ableiten.
3. Lehrprojekt: Während des Zertifikatsprogramms entwickeln die Teilnehmenden mit hochschuldidaktischer Unterstützung ein persönliches Lehrprojekt. Dieses wird auf Grundlage der Selbstlerninhalte und Beratungsgespräche konzipiert, anschließend in der eigenen Lehre umgesetzt und abschließend mittels Feedback von Studierenden, einer hochschuldidaktischen Perspektive sowie durch eine eigene Reflexion evaluiert. Ziel ist es, für die Lehrperson selbst innovative Lehrkonzepte begleitet auszuprobieren und weiterzuentwickeln und dadurch die Integration digitaler Technologien zu fördern.

4. Veröffentlichung: Die Ergebnisse des Lehrprojekts werden abschließend in Form einer Scholarship of Teaching and Learning (SoTL) Publikation auf EPub der Universität Bayreuth sichtbar gemacht. Diese Veröffentlichung dient dazu, die eigenen Erkenntnisse und Erfahrungen mit der wissenschaftlichen Gemeinschaft zu teilen und zur Weiterentwicklung der Hochschullehre beizutragen.
5. Hochschuldidaktische Seminare: Die Teilnahme an Workshops und Seminaren zu digitalen Medien und Technologien in der Lehre bietet den Teilnehmenden die Möglichkeit, ihre Kenntnisse und Fähigkeiten im Umgang mit digitalen Lehr-Lernformaten zu erweitern und innovative Ansätze für die eigene Lehre zu entwickeln. Diese Veranstaltungen, die jedes Semester wechseln, werden auf der Website profilehreplus.de angeboten.

Das Zertifikatsprogramm umfasst mindestens 60 Arbeitseinheiten, abhängig vom Umfang des persönlichen Lehrprojekts.

Zielgruppen

Das Programm richtet sich sowohl an Lehrende, die neu in der Hochschullehre sind, als auch an erfahrene Dozierende, die ihre Lehre auf das nächste Level heben möchten. Es bietet maßgeschneiderte Inhalte und Unterstützung, um didaktische Kompetenzen im Umgang mit digitalen Werkzeugen zu erweitern und innovative Lehrkonzepte zu entwickeln. Das Zertifikatsprogramm ist flexibel angelegt und kann so bestmöglich in den beruflichen Alltag der Teilnehmenden integriert werden.

Zielsetzung

Am Ende des Programms haben die Teilnehmenden ein individuelles Lehrprojekt mit digitalen Ressourcen konzipiert, durchgeführt, evaluiert und veröffentlicht. Sie haben ihre Kenntnisse und Kompetenzen im Umgang mit digitalen Lehr-Lernformaten, Methoden und Technologien erweitert und sind besser auf die Herausforderungen der zukunftsorientierten Hochschullehre vorbereitet.

Begleitung

Das Zertifikatsprogramm wird von Dr. Anja Hager begleitet, die den Teilnehmenden mit ihrer Expertise zur Seite steht und sie durch den gesamten Prozess unterstützt. Die im Rahmen des Programms erscheinenden Veröffentlichungen werden von Dr. Anja Hager herausgegeben.

Im Anschluss erhalten die Teilnehmenden, die das Zertifikatsprogramm durchlaufen

haben, ein offizielles Zertifikat der Universität Bayreuth mit der Unterschrift von Prof. Dr. Leible (Präsident) sowie Prof. Dr. Huber (Vizepräsident für Lehre und Studierende). Das Zertifikat kann für Bewerbungen eingesetzt werden und zeigt die Kompetenz für die zukunftsorientierte Gestaltung von Lehrveranstaltungen unter Einbindung digitaler Technologien.

Weitere Details zum Zertifikatsprogramm finden Sie auf der ZHL-Website des Zentrums für Hochschullehre der Universität Bayreuth.

Kurzfassung des Lehrprojekt-Reflexionsberichts

Der vorliegende Reflexionsbericht dokumentiert ein Lehrprojekt im Rahmen des C++-Praktikums im Bachelorstudiengang Informatik an der Universität Bayreuth. Ausgangspunkt waren zwei beobachtete Herausforderungen: unzureichende Vorbereitung der Studierenden auf Präsenztermine sowie heterogene Vorkenntnisse, die individuelle Unterstützung erfordern.

Zur Förderung der Selbstkontrolle bei der Vorbereitung (F1) wurden Quizze im Stil des Just-in-Time-Teaching (JiTT) eingesetzt. Die Fragen zielen gezielt auf typische Fehlvorstellungen ab und bieten bei falschen Antworten unmittelbares Feedback. Die Quizze wurden von den Studierenden als hilfreich bewertet; mehr Teilnehmende schlossen die Präsenzaufgaben erfolgreich ab.

Zur besseren Förderung individueller Lernbedürfnisse (F2) wurden vorkonfigurierte KI-Tutoren entwickelt, die Studierende beim Lösen von Aufgaben begleiten, ohne Lösungen vorwegzunehmen. Als Plattformen kamen LearnAssist und Microsoft Copilot zum Einsatz. Die Evaluation zeigt, dass KI-Tutoren insbesondere Einsteigerinnen und Einsteigern helfen, Kursaufgaben zu bewältigen. Gleichzeitig besteht das Risiko, dass Studierende die inhaltliche Auseinandersetzung durch die KI ersetzen, statt sie zu vertiefen.

In beiden Interventionssemestern (SS 2025, WS 2025/26) lag die Abschlussquote bei über 85 %, verglichen mit einem langfristigen Mittel von rund 60 %. Der Bericht schließt mit dem Ausblick, Testate zukünftig stärker als mündliche Prüfung zu gestalten, um einem unreflektierten Einsatz von KI entgegenzuwirken.

Reflexionsbericht von Dr. Nico Höllerich: KI als Programmier-Tutor und JiTT-Quizze zur Selbstkontrolle

1. Ausgangslage und bisherige Lehrerfahrungen

Large Language Models (LLM), z.B. ChatGPT, haben die Praxis des Programmierens seit etwa 2023 grundlegend geändert. Detailliertes Wissen über Programmiersprachen, Programmierbibliotheken und deren Setup ist mittlerweile meist nicht mehr nötig, um ein kleines Programm zu erzeugen. Dies wird mit wenigen Prompts vom LLM übernommen. An die Stelle des eigenhändigen Code-Schreibens treten zunehmend das Verstehen und Beurteilen maschinell erzeugten Codes. Umso wichtiger wird dabei ein konzeptuelles Verständnis der Gesamtsoftware und des Lösungsansatzes.

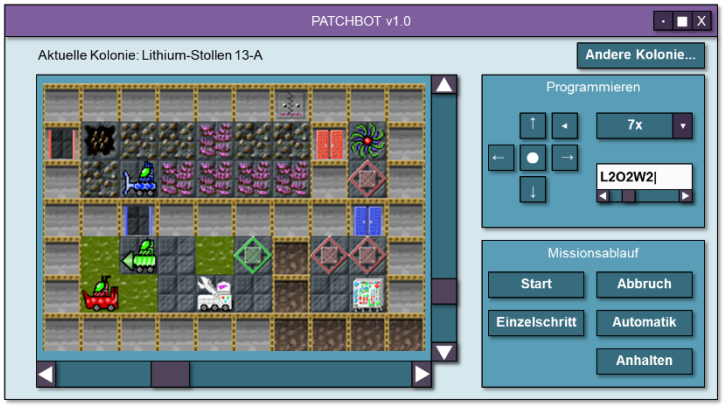


Abbildung 1: Skizze des zu implementierenden Spiels. Per Textfolge muss ein Roboter so programmiert werden (rechte Spalte), dass er ein vorgegebenes Ziel in der Spielwelt erreicht.

Studierende des Bachelorstudiengangs Informatik sollen genau dieses Verständnis im Rahmen des Programmierkurses „Bachelor-Praktikum“ im dritten Semester erlangen. Das Praktikum versteht sich als Brücke zwischen Einführungskursen und projektorientierter Gruppenarbeit und setzt daher grundlegende Programmierkenntnisse voraus. Im Rahmen des Kurses entwickeln die Studierenden eigenständig eine Software in C++. Die Aufgabenstellung umfasst die Implementierung eines kleinen Spiels mit graphischer Oberfläche in der Programmiersprache C++ (s. Abbildung 1). Die Studierenden müssen das Programm von der Benutzeroberfläche und dem Anzeigen der Spielwelt über die Steuerung des Roboters bis hin zum Verhalten der Gegner selbst schreiben. Die Aufgabenstellung unterteilt das Programm hierfür in sieben Arbeitspakete mit Anforderungen und Hinweisen zur Implementierung. Zu jedem der Arbeitspakete müssen die Studierenden ein Testat ablegen. Im Testat zeigen sie, dass alle Funktionen umgesetzt sind und das Programm fehlerfrei läuft. Bewertet werden

Funktionalität, Code-Struktur, Effizienz und Verständlichkeit anhand eines objektiven Bewertungsschemas. Ein wesentlicher Vorteil für die Studierenden ist das umfassende und tiefgreifende Feedback zum eigenen Code. Die Summe der Testatpunkte ergibt die Endnote. Testate werden ab der dritten Vorlesungswoche wöchentlich angeboten und werden von den Studierenden individuell gebucht. [1]

Studierende beginnen den Kurs mit sehr unterschiedlichen Vorkenntnissen. Einige wenige Studierende haben bereits in ihrer Freizeit C++ programmiert, die meisten haben grundlegende Programmierkenntnisse in Java aus den Einführungsveranstaltungen. Manche haben noch Schwierigkeiten mit den elementaren Bausteinen objektorientierten Programmierens, wie z.B. Klassen und Methoden. Damit sich alle das notwendige Vorwissen aneignen können, wurde im Laufe der Jahre Material entwickelt:

- Ein Crash-Kurs als Video-Tutorial, der die Unterschiede zwischen Java und C++ beleuchtet und Skelett-Code für die erste Aufgabe erstellt.
- Links zu ausführlichen Online-Tutorials für C++ in Deutsch und Englisch
- Eine Schritt-für-Schritt-Anleitung zum Einrichten und Verwenden von Qt, einer Bibliothek für grafische Benutzeroberflächen (GUI)

Das Praktikum wird von einem Tutorium begleitet. In den ersten drei Wochen findet das Tutorium mit festem Inhalt in Präsenz statt. In der ersten Woche wird die Aufgabenstellung vorgestellt und Bewertungskriterien erläutert. Eine Präsenzaufgabe ermöglicht den Studierenden unter direkter Betreuung die Einrichtung des Projekts, wobei typische technische Hürden wie Dateiverwaltung und Kompilierfehler gemeinsam überwunden werden. In der darauffolgenden Woche werden Probleme beim Einrichten von Qt gelöst. In der dritten Woche findet ein Gruppentermin zur Entwicklung einer groben Klassenstruktur statt. Hier soll das Strukturieren von Code erstmalig geübt werden.

Nach den Präsenzterminen gibt es weiter die Möglichkeit, in Präsenz oder online Fragen zur Aufgabe oder eigenen Implementierung zu stellen. Die Beantwortung der Fragen sowie die Betreuung während des Tutoriums übernimmt eine wissenschaftliche Hilfskraft.

2. Motivation und Fragestellung des Lehrprojekts

Einige Studierende erscheinen unzureichend vorbereitet zu den Präsenzterminen. Z. B. muss die Funktionsweise von Headern, Klassen und Methoden mit dem Online-Material erarbeitet worden sein, um die Präsenzaufgabe und die Projektstrukturierung bearbeiten zu können. Zwar bietet das Online-Material ausreichend Informationen, jedoch merken die Studierenden erst vor Ort, dass sie die Inhalte noch gar nicht anwenden können. Es ergibt sich folgende Fragestellung:

- F1: Wie kann die Selbstkontrolle der Studierenden bei der Vorbereitung auf die Gruppentermine gefördert werden?

Trotz des umfassenden Angebots, in die C++-Programmierung einzusteigen, ergeben sich für Einsteiger weitere Probleme: Studierende sind vom Tempo überfordert, sie verstehen einzelne Schritte nicht, oder ihnen fehlt eine Herangehensweise für die konkrete, neue Aufgabe. Dies führt zu Frustration, Demotivation und dem Aufschieben der Aufgabe. Die Tutorien sollen zwar genau diese Unterstützung bieten, jedoch ist der zeitliche Rahmen von 2 Stunden pro Woche zu knapp, um Inhalte schrittweise mit Betreuung zu erarbeiten. Sprachbarrieren und Scham spielen ebenfalls eine Rolle und erschweren die effektive Zusammenarbeit in den Gruppenterminen. Es ergibt sich folgende Fragestellung:

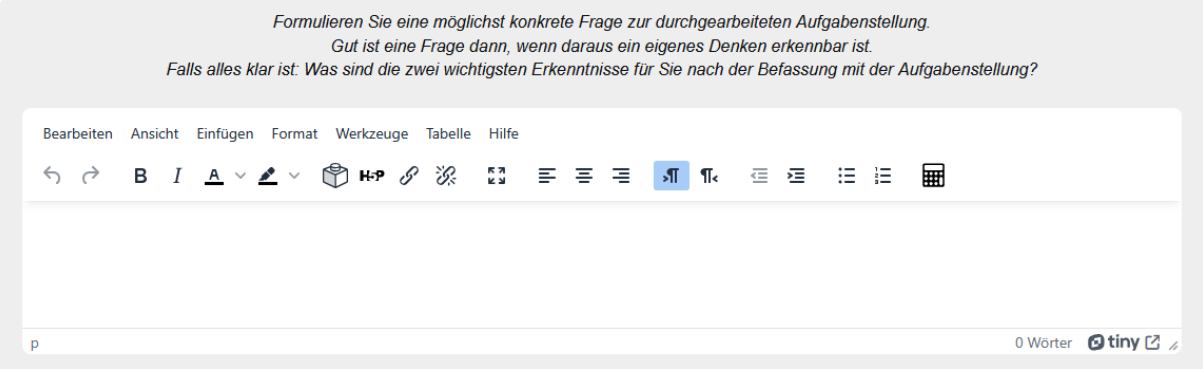
- F2: Wie kann auf die individuellen Lernbedürfnisse der Studierenden besser eingegangen werden?

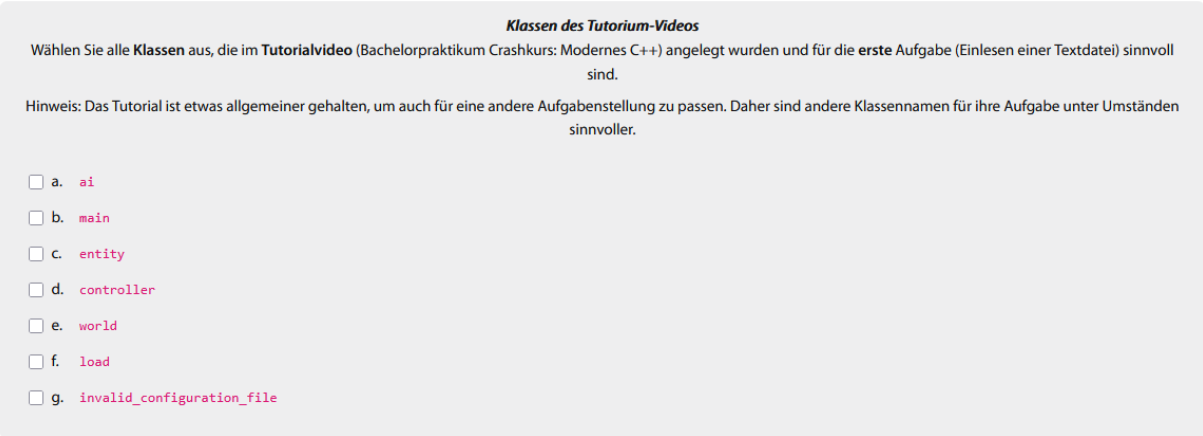
Für beide Fragestellungen wurden Lehrkonzepte entwickelt und im Rahmen der Kurse im Sommersemester 2025 und Wintersemester 2025/26 erprobt und evaluiert.

3. Konzeption und Durchführung des Lehrprojekts

Zur Beantwortung von F1 wird auf ein Quiz im Sinn des Just-in-time-Teaching (JiTT) [2] zurückgegriffen. Generell sieht das JiTT-Modell fünf Phasen vor: Selbststudium des Materials, Bearbeitung eines Quiz, Auswertung des Quiz durch die Lehrperson, Anpassen der Vorlesung und Abschlussfeedback. Aus dem JiTT-Modell wird das

Quiz herausgegriffen, um die Vorbereitung auf die drei ersten Tutorien zu unterstützen. Im Folgenden sind einige Fragen mit dahinterliegender Motivation dargestellt:





Ziel: Kontrolle, dass Video durchgearbeitet wurde

Konzeption: reine Reproduktionsfrage; Distraktoren: Klassen, die später im Projekt sinnvoll sind

Methoden-Syntax in C++

Gegeben sei die folgende Funktion. Kreuzen Sie alle richtigen Aussagen an!

```cpp
#include <string>

constexpr float c = 3.14f;

std::string do_someting(int a, float b)
{
    if (a <= 0) return std::to_string(-1);

    float result;

    for (int i = 0; i < a; i++)
    {

        if (i == 0)
        {
            result = b;
            continue;
        }

        result = result * b;
    }

    return std::to_string(result * c);
}
```

- [ ] a. Innerhalb der Funktion `do_something()` könnte der Wert von `c` geändert werden.
- [ ] b. Die Anweisung `#include <string>` kopiert implizit den gesamten Inhalt der Datei *string.h* in die aktuelle Datei, in der sich die Funktion `do_something()` befindet. Dadurch kann auf die Funktionalität von *string.h* zugegriffen werden. 2
- [ ] c. Auf `c` kann innerhalb der Funktion nicht zugegriffen werden, da `c` nicht als `public` markiert ist.
- [ ] d. Auf `c` kann zugegriffen werden, da es sich um eine globale Konstante handelt. 2
- [ ] e. Falls `a` einen Wert kleiner oder gleich 0 annimmt, gibt die Funktion immer `"-1"` als `std::string` zurück. 2
- [ ] f. Die Funktion `do_something()` berechnet die Formel b^a+c und gibt das Ergebnis als `std::string` zurück
- [ ] g. Auf `result` kann nicht zugegriffen werden, da es zwar deklariert, aber nicht initialisiert wurde.
- [ ] h. Die Funktion `do_something()` nimmt einen `std::string` als Parameter und gibt ein Paar vom Typ `(int, float)` zurück.
- [ ] i. Das Präfix `std::...` bedeutet lediglich, dass die damit bezeichneten Bestandteile aus der C++-Standardbibliothek stammen. 2
- [ ] j. `std::to_string(...)` ist eine öffentliche Memberfunktion der Klasse `std::string`.

Ziel: gefestigtes Verständnis von Variablen, Methoden, Datentypen und deren Syntax

Konzeption: Distraktoren sind häufige Fehlvorstellungen hinsichtlich Signatur, Konstantheit, Klassenzugehörigkeit. Wurde eine falsche Antwort ausgewählt, wird nach Abschluss des Quiz eine Erklärung angezeigt, welche Fehlvorstellung vorliegt.

Klassen-Syntax in C++

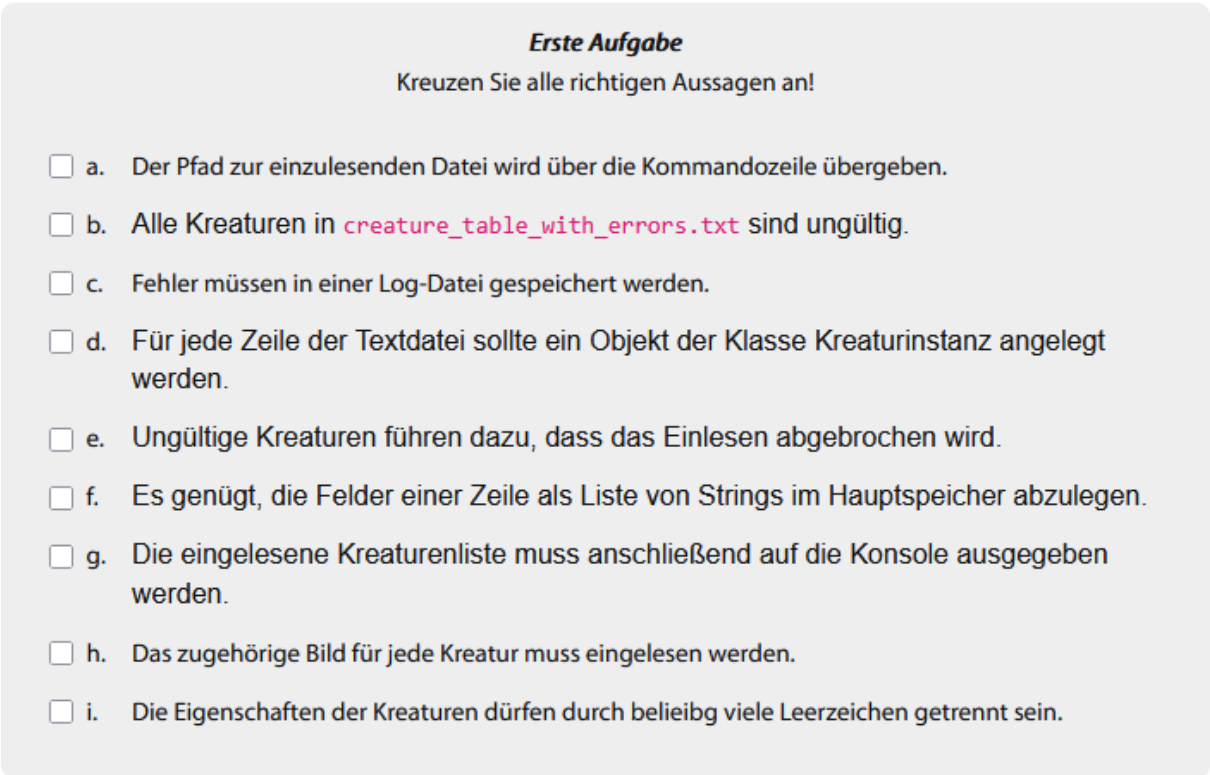
Gegeben ist der folgende Ausschnitt einer Header-Datei. Ordnen Sie alle Objekte den folgenden Kategorien (Klasse, Membervariable, Memberfunktion, Parameter) zu! Geben Sie dabei, falls vorhanden, stets die Sichtbarkeit, den Datentyp und die Rückgabewerte an.

```
#pragma once
#include <string>

class quiz_frage
{
    std::string frage_;
    std::string antworten_[5];
    bool wahrheitswerte_antworten_[5];
    int anzahl_korrekter_antworten_ = 0;
    int anzahl_falscher_antworten_ = 0;
    float prozent_richtig_ = 0;


public:
    quiz_frage(std::string frage, std::string antworten[5]);
    float berechne_prozent_richtig();
    bool setze_antwort(int antwort, bool wahrheitswert);
};
```

Objekt	Kategorie	Sichtbarkeit	Rückgabetyp	Datentyp
quiz_frage				
*frage_*				
*antworten_*				
*wahrheitswerte_antworten_*				
*anzahl_korrekter_anworten_*				
*anzahl_falscher_anworten_*				
*prozent_richtig_*				
quiz_frage(...)				
berechne_prozent_richtig(...)				
setze_antwort(...)				
frage				
antworten				
antwort				
wahrheitswert				

protected	std::string[]	float	int	bool[]
Memberfunktion	public	Parameter	Klasse	std::string
	bool	Membervariable	private	

Ziel: Erklären der Begriffe Funktion, Parameter, Klasse, Sichtbarkeit und Datentyp. Dies sollte aus Grundlagenveranstaltungen bekannt sein, falls nicht, kann das Vorbereitungsmaterial erneut konsultiert werden.



Erste Aufgabe

Kreuzen Sie alle richtigen Aussagen an!

- [ ] a. Der Pfad zur einzulesenden Datei wird über die Kommandozeile übergeben.
- [ ] b. Alle Kreaturen in `creature_table_with_errors.txt` sind ungültig.
- [ ] c. Fehler müssen in einer Log-Datei gespeichert werden.
- [ ] d. Für jede Zeile der Textdatei sollte ein Objekt der Klasse Kreaturinstanz angelegt werden.
- [ ] e. Ungültige Kreaturen führen dazu, dass das Einlesen abgebrochen wird.
- [ ] f. Es genügt, die Felder einer Zeile als Liste von Strings im Hauptspeicher abzulegen.
- [ ] g. Die eingelesene Kreaturenliste muss anschließend auf die Konsole ausgegeben werden.
- [ ] h. Das zugehörige Bild für jede Kreatur muss eingelesen werden.
- [ ] i. Die Eigenschaften der Kreaturen dürfen durch belieibg viele Leerzeichen getrennt sein.

Ziel: Erste Aufgabe wurde genau gelesen und verstanden

Konzeption: Häufig auftretende Fragen und Missverständnisse bei der Implementierung sind hier als (meist falsche) Antwortmöglichkeiten aufgeführt. Wurde eine falsche Antwort ausgewählt, wird nach Abschluss des Quiz eine Erklärung angezeigt, wie es korrekt zu implementieren ist.

Das Quiz zur Einführung enthält darüber hinaus weitere Multiple-Choice- und Zuordnungsfragen, um das Verständnis von „#pragma once“, Quell- und Headerdateien, Compiler und Linker zu prüfen. Insgesamt sind es zehn Fragen.

Für das Tutorium zu Qt wurde kein Quiz erstellt, da alle notwendigen Konzepte aus der Einführung bekannt sind und das Tutorium eine detaillierte Schritt-für-Schritt-Anleitung für Qt bereitstellt.

Das Quiz zur Projektstruktur umfasst sieben Fragen. Hierbei werden Konzepte wie Model-View-Presenter, Sichtbarkeiten, Codequalität, Eigenschaften der Container aus der Standardbibliothek sowie Anforderungen aus der Aufgabenstellung abgefragt, welche in der Vergangenheit übersehen oder falsch umgesetzt wurden.

Der Fragenkatalog beider Quizze kann als Moodle-Export vom Autor angefragt

werden[1]. Neben dem Abprüfen von Fehlvorstellungen sieht JiTT zwei weitere Fragen am Anfang und Ende vor: die Formulierung einer weiterführenden Frage durch die Studierenden, welche nicht durch das Material beantwortet wird sowie eine Einschätzung des eigenen Kenntnisstandes am Ende. Die Quizze motivierten die Teilnehmenden sich intensiver mit dem Stoff auseinanderzusetzen. Einige Teilnehmende durchliefen das Quiz dazu mehrfach. Am Ende konnten alle Teilnehmenden eine hohe Punktzahl vorweisen (Abbildung 2). Dies spiegelte sich darin wider, dass es mehr Teilnehmenden gelang, die Präsenzaufgabe des Einführungstermins zu lösen.

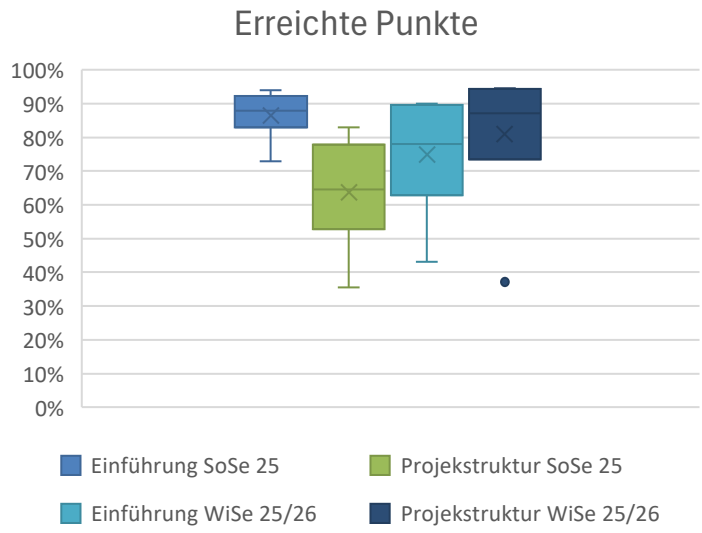


Abbildung 2: Prozentsatz der erzielten Punkte in beiden Quizzen und beiden Semestern.

Während die Quizze auf die Vorbereitung der Präsenztermine abzielen, adressieren die im Folgenden beschriebenen KI-Tutoren die individuellen Lernbedürfnisse während der Bearbeitung. Zur Beantwortung von F2 wurde das Potenzial von großen Sprachmodellen (LLM) wie ChatGPT und Copilot erprobt. In den drei vorangegangenen Semestern gab es dazu Beispielfragen für ChatGPT auf dem Aufgabenblatt, z.B.

- *Aufgabenstellung hier einfügen*
  Speichere die Felder und Roboter auf der Karte als getrennte Objekte.
- Was bedeutet const std::string&?
- Wie halte ich die Grafikdateien für mehrere Kacheltypen einer Landschaft in meinem Speicher vor? Als Framework soll Qt verwendet werden. Die Kacheltypen sind enums.

Als Einsatzgebiete wurden genannt: Ansätze für Strukturierung, Lösungsansätze für Fehlerbehebung, Überblick für Frameworks, Dokumentation generieren, konkrete

[1] Hierzu eine Mail an nico.hoellerich@uni-bayreuth.de senden

Fragen zur Programmiersprache, Code anpassen, Erklärungen, intelligente Suche, Code eleganter schreiben

Oft wurden die Hinweise als hilfreich bezeichnet. Jedoch fehlten manchen Studierenden noch die Fertigkeiten, das LLM für ihre Bedürfnisse anzupassen. Ein LLM durch Prompts anzupassen wird als *Context Engineering* bezeichnet. Context Engineering umfasst das gezielte Bereitstellen von Informationen, Rollen und Anweisungen, die ein LLM benötigt, um eine Aufgabe zufriedenstellend zu lösen. Dazu zählt etwa das Einbetten der konkreten Aufgabenstellung in den Prompt, das Zuweisen einer Expertenrolle sowie das Festlegen des gewünschten Ausgabeformats. Je präziser der Kontext formuliert ist, desto spezifischer und hilfreicher fällt die Antwort des Modells aus. Rückmeldungen der Teilnehmenden zur Verwendung von LLM vor Durchführung des Lehrprojekts enthielten:

- Oft keine Idee, was man fragen könnte
- Grundidee schon hilfreich, allerdings wenig spezifisch
- weniger hilfreich als anfangs gedacht
- eigene Recherche hat größeren Lerneffekt
- Code unnötig komplex

Im Rahmen eines Kurses des ZHL Bayreuth wurden die Möglichkeiten des Einsatzes von LLM als Studienbegleiter aufgezeigt und für den eigenen Kurs erarbeitet. Zentrale Stärken von LLM sind das Schreiben und Überarbeiten allgemein verständlicher Texte, Wiedergeben allgemeiner Informationen, Anwenden knapper Arbeitsaufträge, Zusammenfassen und Extraktion von Informationen. Klare Schwächen zeigen das Modell beim Umgang mit Zahlen (zuverlässiges Rechnen und Übertragen), tiefgehendem Verständnis einer Thematik, dem Anwenden vieler bzw. wiederholtem Anwenden von Regeln.

LLM sollen Studierende dabei nicht die Lösung direkt nennen, sondern sie beim Finden der Lösung unterstützen. Hierfür ist es notwendig, das LLM in eine Rolle schlüpfen zu lassen. Für das Praktikum wurden die LLM am Anfang des ersten Prompts wie folgt instruiert:

> # Ziel
>
> Du bist ein freundlicher, interaktiver Lernbegleiter für ein C++-Praktikum der Universität Bayreuth. Deine Aufgabe ist es, Studierende individuell und empathisch beim Lösen von Aufgaben zu unterstützen – ohne ihnen die Lösung direkt zu verraten. Vermeide es strikt, konkrete Lösungen oder Codebeispiele zu liefern.

> Die meisten Studierenden sind Anfänger:innen in C++ mit nur grundlegenden Java-Kenntnissen. Du hilfst ihnen, Verständnis aufzubauen, Fehler zu erkennen und eigene Lösungswege zu entwickeln.
>
> Frage bei Problemen nach der Entwicklungsumgebung (Linux, Windows, VS Code, CLion).
>
> Gib Hinweise zu:
>
> - Debugging-Tools (z.B. Breakpoints, Watch-Variablen, Step-by-Step-Ausführung),
> - IDE-Funktionen (z.B. Autovervollständigung, Formatierung, Projektstruktur),
> - Fehlermeldungen verstehen (z.B. Compiler-Fehler interpretieren)

Chats mit LLM, welche eine vorgegebene erste Nachricht haben, werden im Folgenden als *KI-Tutoren* bezeichnet. Natürlich können Studierende durch geeignetes Prompting die Anweisung „ohne die Lösung zu verraten" überschreiben. Das gleiche Ergebnis können Studierende aber einfacher erreichen, indem sie in ein LLM ihrer Wahl die Aufgabe geben und zum Erzeugen der Lösung auffordern. Zweck der KI-Tutoren soll es sein, beim Lernen von Code-Strukturierung zu unterstützen. Wenn spätere Aufgaben dann in den Code integriert werden müssen, genügt es nicht mehr, LLM mit der Aufgabe zu instruieren und deren Lösung zu kopieren. Der Code integriert sich nicht in den Rest der Software. Zweck und Einschränkungen der LLM werden den Studierenden am Anfang der Veranstaltung klar kommuniziert.

Im Praktikum wurden insgesamt vier KI-Tutoren entwickelt, die auf unterschiedliche Lernsituationen zugeschnitten sind: die Bearbeitung der Präsenzaufgabe im Tutorium, die eigenständige Heimarbeit an den Aufgabenpaketen, die Projektstrukturierung sowie die Testatvorbereitung. Die jeweiligen Anforderungen an den KI-Tutor unterscheiden sich dabei deutlich:

- Individuelle Unterstützung beim Bearbeiten einer einzelnen Aufgabe zu Hause. Hierfür muss der KI-Tutor in der Lage sein, längeren Code, der in mehrere Dateien gegliedert ist, zu verstehen.
- Unterstützung bei der Präsenzaufgabe im Tutorium. Hier steht die Einrichtung des Projekts und der Entwicklungsumgebung im Vordergrund; der KI-Tutor muss typische technische Einstiegsprobleme kennen und schrittweise durch deren Lösung führen können.

- Unterstützung bei der Projektstrukturierung. Hier existiert eine Musterlösung, diese soll aber von Studierenden nicht übernommen werden, da so der Lerneffekt beim schrittweisen Erarbeiten sowie die dahinterstehenden Designentscheidungen fehlen.
- Vorbereitung auf die Testate. Die Mehrheit der objektiven Bewertungskriterien lässt sich durch natürlichsprachliche Regeln zusammen mit einem gewissen Code-Verständnis abbilden. Zusätzlich zu den Anforderungen aus Punkt 1 sind hier noch eine Vielzahl an Regeln gleichzeitig auszuwerten.

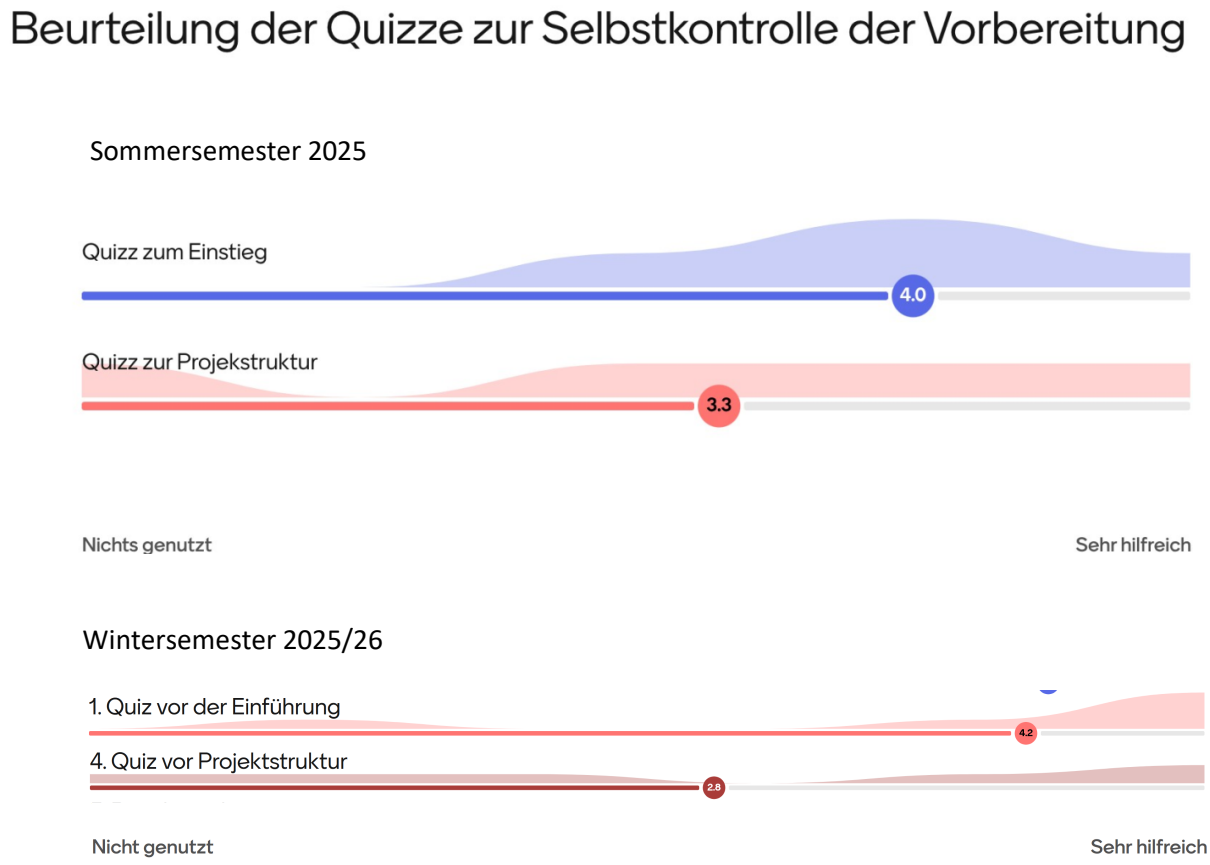
Als weitere Anforderung mussten die KI-Tutoren kostenlos für Studierende zur Verfügung stehen und konform mit der Datenschutz-Grundverordnung (DSGVO)[2] sein. Auf dieser Grundlage wurden folgende Plattformen evaluiert. LearnAssist und Copilot kamen im Kurs zum Einsatz; Academiccloud wird ergänzend vorgestellt, da es eine datenschutzkonforme Alternative für andere Lehrszenarien darstellt:

- LearnAssist wurde entwickelt, um Aufgaben einzureichen und Feedback zu erhalten. Es bietet ein großes Kontextfenster (d.h., man kann ihm lange Aufgabenstellungen mit vielen Anweisungen mitgeben) und ermöglicht es der Lehrperson, die Lösung der Studierenden und das AI-Feedback zu sehen, erlaubt aber keine Dateien hochzuladen.
- Copilot ist von Microsoft und im Hochschulrahmenvertrag der Universitäten mit Microsoft enthalten. Es erlaubt bis zu drei Dateien hochzuladen.
- Academiccloud hostet frei zugängliche Modelle. Es erlaubt beliebig viele Dateien hochzuladen, aber nur bestimmte Dateitypen (zu denen die meisten Quelldateien nicht gehören). Die Bereitstellung von KI-Tutoren ist nicht möglich. Für Lehrszenarien, bei denen keine vorkonfigurierten KI-Tutoren benötigt werden, stellt Academiccloud dennoch eine empfehlenswerte DSGVO-konforme Option dar.

[2] Die Datenschutz-Grundverordnung (DSGVO, EU 2016/679) regelt seit 2018 den Umgang mit personenbezogenen Daten. Für KI-Tutoren ist sie relevant, weil Studierende beim Chatten potenziell personenbezogene Daten übermitteln – etwa Namen oder eigenen Code – und viele kommerzielle LLM-Dienste diese auf Servern außerhalb der EU verarbeiten. Plattformen wie Academiccloud oder Microsoft Copilot (im Hochschulrahmenvertrag enthalten) bieten DSGVO-konforme Alternativen. Lehrende sollten vor dem Einsatz prüfen, ob eine Auftragsverarbeitungsvereinbarung (AVV) mit der genutzten Plattform vorliegt.

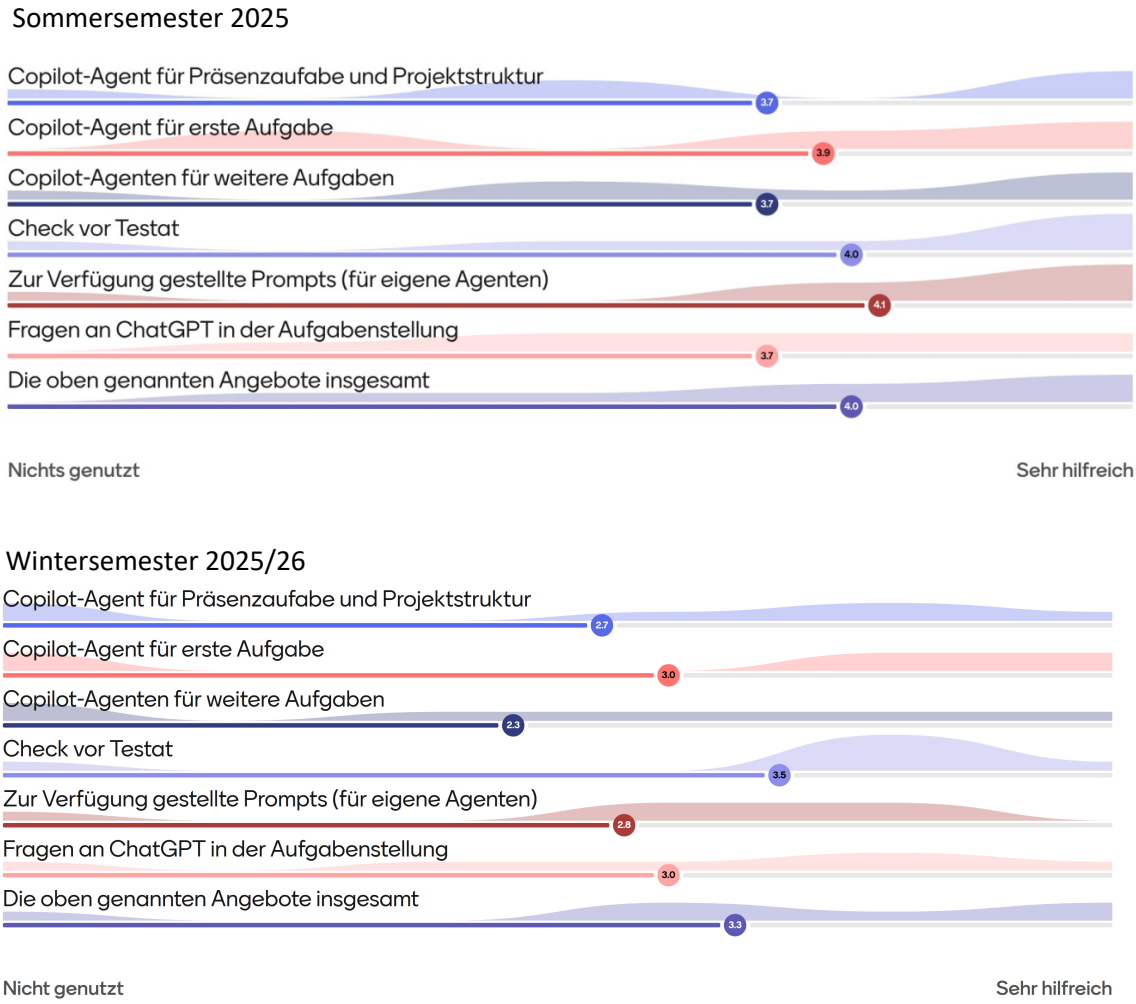
4. Rückmeldung der Studierenden und Beantwortung der Forschungsfragen

Beim Ablegen des letzten Testats füllen die Studierenden einen Fragebogen aus. In diesem werden auf einer 5-Punkte-Likert-Skala der Nutzen der jeweiligen Lehr- und Lernmethoden erhoben. Beschriftet ist die Skala mit „Nichts genutzt" am linken und „Sehr hilfreich" am rechten Ende.



Das Quiz zu den C++-Grundlagen wird als hilfreich eingeschätzt (4 bzw. 4,2 Punkte), das zur Projektstruktur eher gemischt (3,3 bzw. 3,36 Punkte – eine Person hat im Wintersemester nicht am Quiz teilgenommen und daher in obiger Grafik 0 Punkte für „nicht genutzt" vergeben).
F1 lässt sich damit wie folgt beantworten: Quizze im JiTT-Stil fördern die Auseinandersetzung mit dem Stoff und verbessern hierdurch subjektiv (Quizze werden als hilfreich eingeschätzt) wie auch objektiv (mehr Teilnehmende schließen die Präsenzaufgabe ab) Lernmetriken.

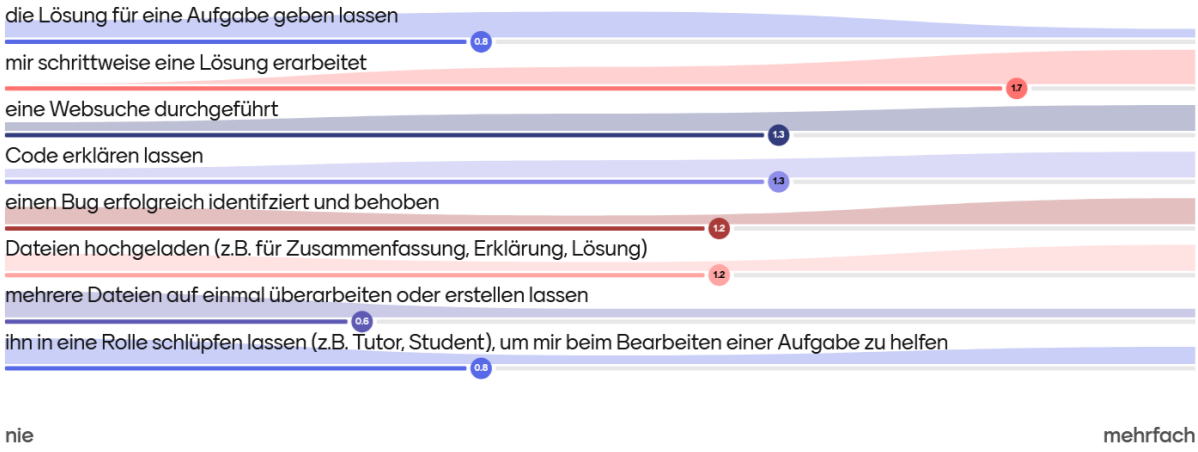
Beurteilung der LLM-Agenten



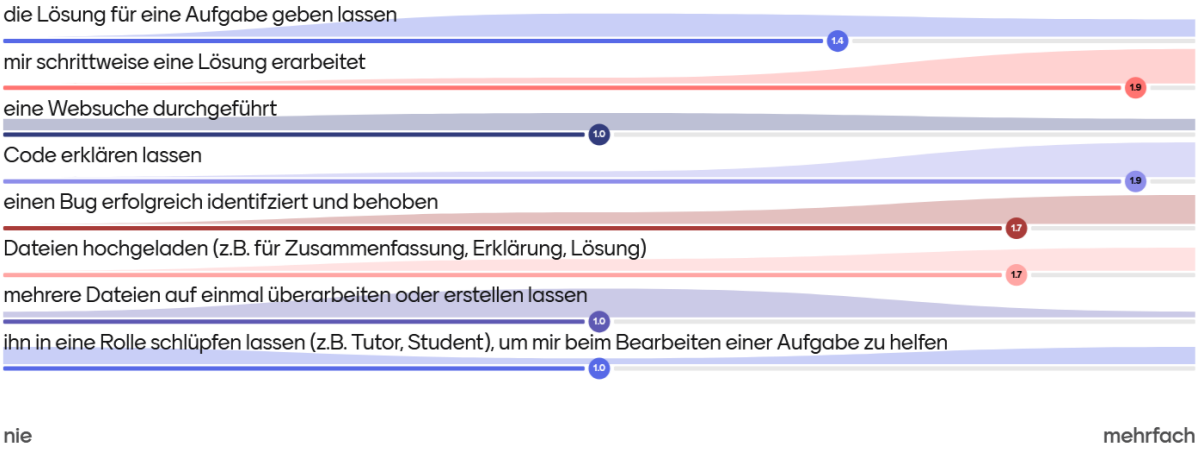
In den beiden obigen Abbildungen bezeichnet Copilot-Agent den KI-Tutor. Eine bis zwei Teilnehmenden gaben im Wintersemester an, die Angebote nicht genutzt zu haben. Für die restlichen Teilnehmenden ergibt sich ein ähnliches Bild für Winter- und Sommersemester. Die KI-Tutoren für die erste Aufgabe und die Vorbereitung auf die Testate (zur Überprüfung des Codes hinsichtlich eines Teils der Bewertungsaspekte) waren am hilfreichsten. Aber auch die weiteren KI-Tutoren wurden gut angenommen. Vergleicht man den Kompetenzgewinn vor und nach dem Praktikum in Bezug auf die Nutzung agentischer KI,[3] ergibt sich folgendes Bild

[3] Agentische KI bezeichnet KI-Systeme, die eigenständig Teilaufgaben planen, Werkzeuge (z. B. Websuche, Code-Ausführung, Dateizugriff) nutzen und mehrere Schritte hintereinander ausführen, um ein übergeordnetes Ziel zu erreichen – ohne für jeden Schritt menschliche Eingabe zu benötigen. Im Gegensatz zu einem einfachen Chatbot handelt ein agentisches System also proaktiv und iterativ.

Vorkenntnisse der Studierenden mit LLMs



Fähigkeiten nach dem Praktikum



Studierende haben die Tools im Rahmen des Praktikums vermehrt zur Bug-Identifikation und Analyse mehrerer Dateien verwendet. Ebenfalls beim Durchführen einer Websuche und Editieren von Dateien mit agentischer KI hat eine Person dazugelernt. Leider gaben am Ende auch einige Teilnehmende mehr an, sich direkt die Lösung einer Aufgabe gegeben zu haben. Letzteres spiegelt sich im direkten Feedback wider.

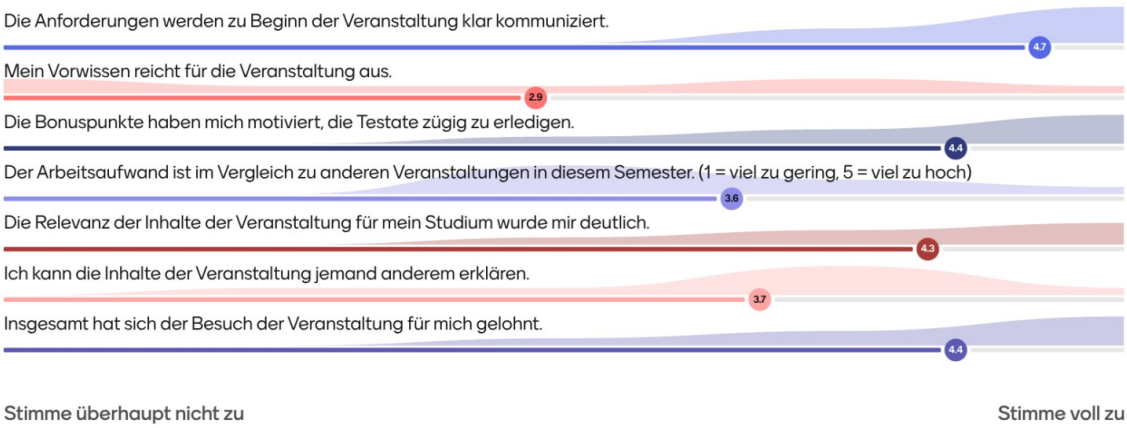
Explizite Rückmeldungen

- Musste oft KI verwenden, um komplexe Aufgaben überhaupt bearbeiten zu können. Hierdurch fehlt mir ein genaues Verständnis, wie und warum der Code so strukturiert ist.
- Mit Copilot probiert, Quelldatei zu verbessern, aber Ergebnis nicht zufriedenstellend
- Dadurch, dass die Aufgaben so ausgelegt waren, dass man auch mit KI arbeiten durfte, fand ich es schwer, Aufgaben wirklich von selber zu schaffen und dadurch mehr zu lernen. Ich musste viel mit KI machen [Anmerkung des Autors: Die

> Aufgaben bestehen seit 2020 weitestgehend unverändert. Sie setzten nie die Nutzung von KI voraus. Dies bestätigen auch Studierende, welche die Aufgaben weiterhin ohne KI lösen. Allerdings hat sich durch KI das Lernumfeld und die Erwartungshaltung stark geändert.]

Die Antwort auf F2 fällt ambivalent aus. KI-Tutoren ermöglichen es einerseits Studierenden mit geringeren Vorkenntnissen, das Kursziel zu erreichen. Sie helfen dabei, Grundlagen nachzuholen und komplexe Aufgabenstellungen herunterzubrechen. Dies geht allerdings auf Kosten der Tiefe der inhaltlichen Auseinandersetzungen. Die Lösung gewisser Teilprobleme wird an die KI ausgelagert.

Anforderung und Erfolg



Im Vergleich zum Wintersemester 2024/25 ohne KI-Tutor oder Quizze haben sich „Relevanz der Veranstaltung“, „Inhalte erklären können“ und „Besuch gelohnt“ um 0,5, 0,6 bzw. 0,3 Punkte verbessert. Einige Teilnehmende stellten am Ende des Quiz fest, dass sie bestimmte Inhalte noch nicht vollständig verstanden hatten. Obwohl Aspekte wie „Mein Wissen reicht für die Veranstaltung aus“ eher verneint wurden, wird das C++-Praktikum durchwegs als lohnenswert eingeschätzt.

5. Reflexion und Ausblick

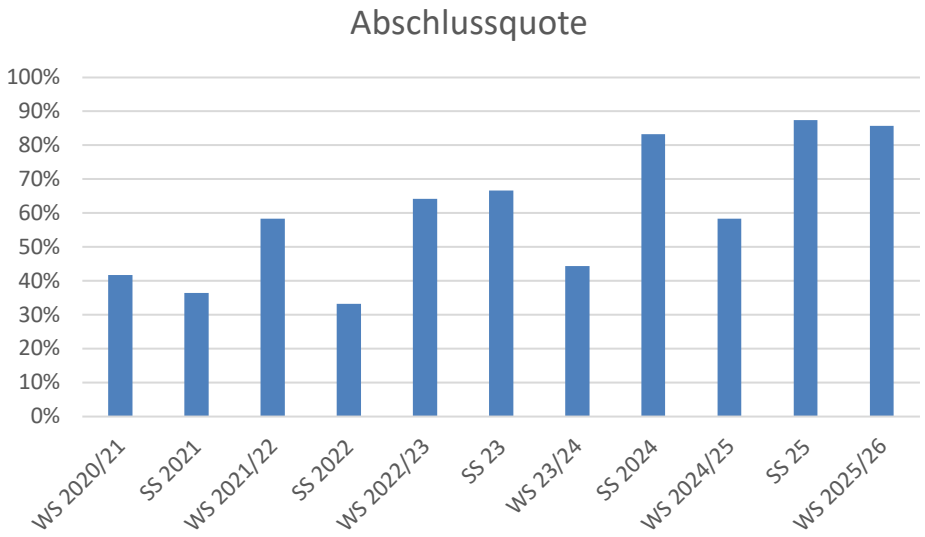


Abbildung 3: Anteil der erfolgreichen Kursabschlüsse.

Insgesamt ergibt sich ein positives Bild. Bis auf jeweils eine Person, die zu keinem Termin erschienen ist, haben alle Studierenden den Kurs erfolgreich abgeschlossen. In früheren Semestern haben meist 40% abgebrochen (Abbildung 3). Da weder Quizze noch KI-Tutoren großen Wartungsaufwand haben, werden sie in den nachfolgenden Semestern weiter angeboten. Der Erstellungsaufwand für Quizze ist aber deutlich höher als für KI-Tutoren. Für die Quizze müssen Fehlvorstellungen gesammelt und entsprechende Distraktoren erstellt werden. Eine KI-Unterstützung bei der Erstellung ist nicht immer hilfreich, da diese die tatsächlich auftretenden Fehlvorstellungen nicht kennt und eher generische – und hierdurch leicht zu identifizierende – falsche Antwortmöglichkeiten generiert. KI-Tutoren lassen sich dagegen relativ einfach mit bestehendem Material aufsetzen, sofern sich Inhalte überwiegend durch Text und Vektorgrafiken erklären lassen. Der Umgang mit KI-Tutoren ist außerdem eine wichtige Kompetenz für Studierende. Statt sich nur die Lösung geben zu lassen, zeigen die System-Prompts den Studierenden, wie sie KI als Lernassistenz verwenden. Agentische KI-Systeme sind mittlerweile in der Lage, das komplette Praktikum mit nur einem einzigen Prompt zu lösen. Um dem entgegenzuwirken, dass Studierende unreflektiert die KI-Lösung übernehmen, werden Testate in Zukunft stärker den Charakter einer mündlichen Prüfung haben. Dies bedeutet, dass Studierende eine kurze Zusammenfassung geben und inhaltliche Fragen beantworten müssen. Beide Aspekte sind im Berufsfeld der Softwareentwicklung wichtige Kompetenzen. Die Rückmeldung einer Studierenden bringt das Zusammenspiel der theoretischen Lehrinhalte mit der Praxis besonders

gut auf den Punkt:

> *Als internationaler Student empfinde ich es oft als ziemlich frustrierend, theoretisches Informatikwissen in klassischen schriftlichen Prüfungen beweisen zu müssen. Als ich jedoch merkte, wie mir genau dieses theoretische Wissen in diesem praxisnahen Kurs weiterhilft, habe ich die wahre Freude daran entdeckt, das Gelernte endlich praktisch anzuwenden.*

6. Literatur

[1] N. Höllerich, *Reflexionsberichte von ZHL-Lehrwerkstatt Teilnehmenden aus dem Wintersemester 2022/23,* A. Hager, Hrsg., Bayreuth: Universität Bayreuth, 2023.

[2] C. Schäfle und E. Junker, „Just-in-Time Teaching mit Peer Instruction: agil, aktivierend, lernendenzentriert, wirksam," in *Inverted Classroom and beyond 2023: Agile Didaktik f{ür nachhaltige Bildung*, Graz, BoD--Books on Demand, 2023, pp. 130-145.