die hochschul*lehre*
Interdisziplinäre Zeitschrift für Studium und Lehre

die hochschul*lehre* – Jahrgang 12 – 2026 (18)
Herausgebende des Journals: Svenja Bedenlier, Ivo van den Berk, Sarah Berndt, Jonas Leschke, Peter Salden, Antonia Scholkmann, Angelika Thielsch
Beitrag in der Rubrik Praxisforschung
DOI: 10.3278/HSL2618W
ISSN: 2199–8825 wbv.de/die-hochschullehre



# KI-unterstützte Programmierung mittels ergebniszentrierter Fehlerklassifikation

## *Potenziale zur Schaffung neuer Lernräume*

Sebastian Stemmler, Jens Ahlers & Robert Göllinger

**Zusammenfassung**

Programmierkenntnisse gewinnen in technischen Studiengängen zunehmend an Bedeutung. Jedoch stehen in Programmierübungen häufig zu wenige Tutor:innen zur Verfügung, die die Studierenden beim Lösen der Programmieraufgaben unterstützen. Deshalb wird ein KI-System entwickelt, das die Studierenden jederzeit und individuell beim Lösen der Programmierübungen unterstützt. Im Gegensatz zu klassischen Ansätzen analysiert das KI-System nicht den programmierten Code, sondern fokussiert sich auf die Zwischen- und Endergebnisse (z. B. erzeugte Daten, Plots). Anhand dieser gibt das KI-System Hinweise und Erklärungen, um das selbstständige Arbeiten und kritische Denken ohne Musterlösung zu fördern. Ein erster Prototyp wurde im Rahmen einer Programmierübung eingesetzt und mit positiver Resonanz evaluiert.

**Schlüsselwörter:** Programmierübung; KI-System; Prototyp; individuelles Lernen; Fehlerklassifikation

## AI-Assisted Programming through Result-Centered Error Classification

### *Potentials for Creating New Learning Rooms*

**Abstract**

Programming skills are becoming increasingly important in technical degree programs. However, there are often too few tutors available in programming exercises to support students in solving programming tasks. Therefore, an AI system is being developed that supports students at any time and individually in solving programming exercises. Unlike traditional approaches, the AI system does not analyze the programmed code but focuses on intermediate and final results (e. g., generated data, plots). The AI system uses these to provide hints and explanations to encourage independent work and critical thinking without a given solution. The first prototype was implemented in the context of a programming exercise and evaluated with positive feedback.

**Keywords:** Programming Exercise; AI System; Prototype; Individual Learning; Error Classification

# 1 Einleitung

In der heutigen digitalen Welt sind Programmierkenntnisse für technische Berufe unverzichtbar geworden, da die Softwareentwicklung ein zentraler Innovationsfaktor zur Entwicklung neuer technischer Systeme ist. Vor diesem Hintergrund gewinnen Programmierübungen in technischen Studiengängen immer mehr an Bedeutung, da sie den Studierenden nicht nur theoretisches Wissen vermitteln, sondern auch praktische Fähigkeiten fördern. Diese Fähigkeiten zählen zu den sogenannten „21st Century Skills", insbesondere zu den Kompetenzen „Information and communications technology operations and concepts", die für die akademische und berufliche Bildung gefordert werden (Ananiadou & Claro, 2009). Ein unverzichtbarer Bestandteil dieser Programmierübungen sind Tutor:innen, die die Lernenden bei der Bearbeitung der Programmieraufgaben unterstützen. Allerdings stehen diese oft nur während der Präsenzzeit der Übung zur Verfügung. Darüber hinaus betreut aufgrund der begrenzten finanziellen Mittel ein:e Tutor:in nicht selten eine große Anzahl an Studierenden. Auch die Zeit der Tutor:innen wird nicht bestmöglich genutzt, da sie oftmals mit leicht zu beantwortenden und wiederkehrenden Fragen konfrontiert sind. All diese Faktoren führen in vielen Fällen zu einer unzureichenden Betreuung der Studierenden, was den Lernerfolg behindert.

Die derzeitige Lehrpraxis stützt sich zudem auf traditionelle Methoden wie die Bereitstellung von Musterlösungen oder die automatische Codeanalyse, sodass den Studierenden ein definierter Lösungsweg vorgegeben wird. Dies führt dazu, dass dieser Lösungsweg präferiert und somit ein individueller Lösungsweg oft unterbunden wird. Zur Unterstützung und Förderung eines individuellen Lösungsweges wird im Rahmen des Projektes „KI-unterstützte Programmierübung" ein neuartiges KI-System entwickelt, das den Studierenden jederzeit zur Verfügung steht. Dieses KI-System soll im Gegensatz zu heute bekannten Unterstützungssystemen nicht den Code analysieren, sondern sich auf die Bewertung von Zwischen- und Endergebnissen konzentrieren. Hierfür werden die Zwischen- und Endergebnisse mithilfe künstlicher neuronaler Netze klassifiziert und die vorliegenden Fehler ermittelt. Dies ermöglicht dem KI-System, den Studierenden automatisierte Rückmeldungen und Unterstützung zu geben.

Der Einsatz eines KI-gestützten Systems verspricht nicht nur eine Verbesserung des Verständnisses von komplexen Zusammenhängen für die Studierenden, sondern entlastet zudem menschliche Tutor:innen von Routinefragen und ermöglicht ihnen eine konzentrierte Auseinandersetzung mit individuellen Detailfragen. Das KI-System leistet somit einen wichtigen Beitrag zur Weiterentwicklung innovativer Lehrkonzepte im Bereich technischer Studiengänge und zeigt auf, wie moderne Technologien genutzt werden können, um das Lernen effektiver und individueller zu gestalten. Hierfür werden in Abschnitt 2 zunächst die theoretischen und empirischen Grundlagen vorgestellt. Anschließend werden in Abschnitt 3 mögliche Lehrkonzepte und die Funktionsweise des KI-Systems beschrieben. Abschließend werden in Abschnitt 4 die Evaluationsergebnisse vorgestellt und in Abschnitt 5 die gewonnenen Erkenntnisse diskutiert.

# 2 Theoretische und empirische Grundlagen

Mit stetig wachsender Bedeutung der Programmierausbildung rücken Herausforderungen zur Unterstützung von Studierenden sowie dem Einsatz innovativer Technologien wie Künstlicher Intelligenz (KI) stärker in den Fokus der Lehre. In den letzten Jahren hat sich die Programmierausbildung stark verändert, wobei ein zunehmender Fokus auf praxisorientierten Lehrmethoden liegt. Theorien des konstruktivistischen Lernens betonen die Bedeutung aktiver Teilnahme und selbstgesteuerten Lernens (Baker & Siemens, 2014). Programmierübungen haben sich als effektives Mittel etabliert, um nicht nur technische Fähigkeiten zu vermitteln, sondern auch Problemlösungsfähigkeiten zu fördern.

Traditionelle Lehrmethoden stützen sich vor allem auf Analysemethoden zur Bewertung des programmierten Codes. Die derzeitige Lehrpraxis umfasst hierfür vor allem statische oder dynamische Ansätze der Codeanalyse. Die statische Analyse bewertet dabei den Code, ohne ihn auszuführen (Striewe & Goedicke, 2014). Hierdurch können beispielsweise Typendefinition, Datenflüsse (Veränderung der Daten im Programmverlauf) und Kontrollflüsse (Reihenfolge der ausgeführten Operationen) überprüft werden. Im Kontext der Programmanalyse lässt sich zwischen der Analyse des Quellcodes und der Analyse des vom Compiler erzeugten Codes (z. B. Bytecode) unterscheiden. Für Lehrzwecke ist jedoch nahezu ausschließlich die Quellcodeanalyse von Bedeutung, da sie direkt an den von den Studierenden geschriebenen Programmen ansetzt. Mithilfe abstrakter Syntaxbäume kann die Codestruktur komprimiert dargestellt und analysiert werden, sodass auch die Semantik analysiert werden kann (Truong et al., 2005).

Die dynamische Analyse bewertet dagegen das Verhalten eines Programms während seiner Ausführung (de Silva, Samarasekara & Hettiarachchi, 2023). Hier existieren verschiedenste Verfahren, die vorrangig dazu dienen die Leistungsfähigkeit des Codes zu bewerten sowie sicherheitskritische Implementierungsfehler und im Code nicht berücksichtigte Zustände zu identifizieren. Truong et al. (2005) stellen ein Framework vor, das die dynamische Analyse als Schlüsselkomponente einsetzt, um das Verhalten des Programms unter verschiedenen Bedingungen zu evaluieren. Hierdurch können Abweichungen zur Musterlösung identifiziert und den Studierenden als Feedback mitgeteilt werden. Fonte et al. (2013) erweitern die klassische dynamische Analyse, indem sie die Ausgabe des Programms mit der erwarteten Ausgabe auf semantischer Ebene vergleichen.

Ein hybrider Ansatz kombiniert sowohl statische als auch dynamische Analysemethoden und bietet somit eine umfassendere Fehlerabdeckung. Ein Beispiel hierfür ist das Tool ASSYST, das verschiedene Arten von Fehlern identifizieren kann – von syntaktischen bis hin zu logischen Fehlern (Jackson & Usher, 1997). Ein weiterer Ansatz ist AutoLEP, das syntaktische Fehler und strukturelle Mängel des Codes entdeckt (Wang et al., 2011). Außerdem wird bei Ramos et al. (2013) eine Erweiterung des Moodle-Lernmanagementsystems vorgestellt, die MATLAB-Aufgaben automatisch korrigieren kann.

Zunehmend finden auch KI-gestützte Systeme Eingang in die Programmierausbildung. Statt nur den finalen Code zu bewerten, analysieren diese KI-gestützten Systeme auch Zwischenstände, um personalisiertes Feedback zu ermöglichen und individuelle Lösungswege zu fördern. Ein Beispiel ist der Onlinekurs „CS50“ der Harvard University, der KI-basierte Feedbackmechanismen einsetzt, um Studierende bereits während des Programmierens auf typische Fehler oder Verbesserungsmöglichkeiten hinzuweisen (Malan et al., 2021). Hierbei werden vordefinierte Testfälle mit verschiedenen Eingaben ausgeführt und die Ausgaben verglichen.

Im Unterschied zu klassischen Ansätzen können ML-Modelle aus großen Datenmengen lernen und komplexe Muster im Quellcode erfassen (Sharma et al., 2021). Jedoch hängt ihr Erfolg stark von der Qualität der verwendeten Datensätze ab. Die Vorbereitung dieser Datensätze ist zeitaufwendig und erfordert zudem das Training komplexer Modelle mit hoher Rechenleistung.

Alle vorgestellten Ansätze können verschiedenste Fehler identifizieren. Jedoch hängt die Fehlererkennung maßgeblich von der implementierten Musterlösung bzw. deren bereitgestellten Lösungsvarianten ab. In ingenieurwissenschaftlichen Fächern sind zudem visuelle Ausgaben (z. B. Plots, Zeitverläufe und Kennfelder) von großer Bedeutung, da sie den Studierenden helfen, algorithmisches Verhalten zu verstehen und potenzielle Fehler zu erkennen. Dieser Aspekt wird bisher von allen beschriebenen Ansätzen vernachlässigt. Die gezielte Analyse solcher visuellen Ergebnisse bietet jedoch Potenzial, um die Qualität der Betreuung weiter zu verbessern. Diese Lücke adressiert das im Folgenden vorgestellte KI-System, indem es die Analyse von Zwischen- und Endergebnissen integriert und damit eine individuelle Rückmeldung ermöglicht.

# 3 Beschreibung des Lehrkonzeptes

Das KI-System wird im Rahmen der Lehrveranstaltung „Maschinelles Lernen in der industriellen Regelungstechnik“ entwickelt und erprobt. Die Lehrveranstaltung setzt sich zusammen aus einer Vorlesung und einer Übung. In der Vorlesung werden die theoretischen, methodischen Grundlagen vermittelt und in der Übung angewendet. Die Übung erfolgt in Form von Programmieraufgaben in MATLAB/Simulink, um eine ingenieurstypische Arbeitsweise zu lehren. Vorlesung und Übung sind über die Lernziele nach Bloom (1956) miteinander verknüpft, sodass Erlerntes aus der Vorlesung direkt in der Übung angewendet werden kann.

## 3.1 Didaktischer Ansatz

Im Rahmen der Programmierübung sollen die Studierenden zu einer ingenieursnahen und selbstständigen Arbeitsweise motiviert werden. Deshalb werden die Programmierübungen zunächst in Form einer Selbstlernphase vorbereitet. In der anschließenden Präsenzzeit können auftretende Probleme und Fragen mit den Tutor:innen diskutiert werden. Während der Präsenzzeit zeigt sich jedoch, dass die Programmieraufgaben häufig nicht bearbeitet werden. Sei es, weil die Studierenden sich nicht sicher sind, ob ihre Lösung richtig ist, oder weil sie aufeinander aufbauende Teilaufgaben nicht lösen können. Aus diesem Grund ist das KI-System so konzipiert, dass es den Studierenden zu jeder Zeit zur Verfügung steht. Es kann also zur Vorbereitung während der Selbstlernphase, während der Präsenzzeit, zur Nachbearbeitung und zur Klausurvorbereitung genutzt werden. Zudem reduziert das System die Barriere zur Inanspruchnahme von Unterstützung auf ein Minimum, da es anonym von den Studierenden genutzt wird und dadurch keine gefühlten Konsequenzen folgen können. Weiterhin berücksichtigt das KI-System unterschiedliche Leistungsstände und Fähigkeiten der Studierenden. Aus diesem Grund ist das KI-System so realisiert, dass es die folgenden Rahmenbedingungen berücksichtigt:

1. Die Studierenden werden bei der Umsetzung individueller Lösungswege unterstützt, sodass eine Musterlösung nicht benötigt wird.
2. Studierende können selbst entscheiden, ob sie Feedback von dem KI-System erhalten.
3. Studierende erhalten auch nach Teilaufgaben eine Rückmeldung von dem KI-System und nicht erst nach der vollständigen Bearbeitung der Aufgaben.
4. Die Erläuterungen des KI-Systems erfolgen mit unterschiedlichem Detaillierungsgrad. So wird bei erstmaligem Auftreten eines Fehlers nur die Information ausgegeben, dass ein Fehler vorhanden ist. Bei wiederholtem Auftreten des gleichen Fehlers wird die Unterstützung inkrementell ausführlicher.

Hierdurch lässt sich ein holistisches Lehrkonzept gemäß Abbildung 1 realisieren, das Selbstlernphasen und Präsenzphasen als ganzheitlichen Lernprozess betrachtet. Anstatt Wissen nur während der Präsenzzeit zu vermitteln, können Studierende so unterschiedliche Unterstützungsformate und Erklärungsarten in Anspruch nehmen. Es ist keinesfalls das Ziel, dass das KI-System die Tutor:innen vollständig ersetzt, da die menschliche Interaktion und Beobachtung wesentliche Bestandteile der individuellen Förderung und Unterstützung in der Lehre sind.

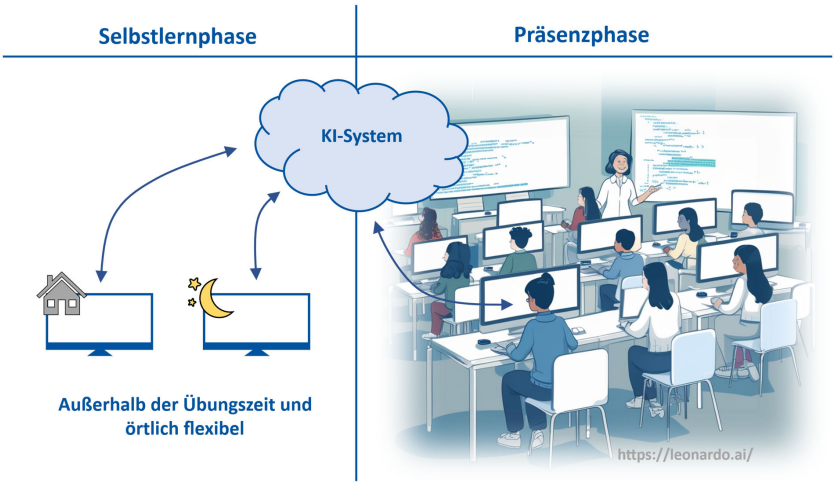


**Abbildung 1:** Angestrebtes didaktisches Konzept bestehend aus Selbstlern- und Präsenzphase, wobei die Interaktion zwischen Studierenden und KI-System jederzeit möglich ist

## 3.2 Funktionsweise des KI-Systems

Im Gegensatz zu klassischen Systemen wird nicht der programmierte Code bewertet. Stattdessen sollen Zwischen- und Endergebnisse in Form von Daten, Plots etc. analysiert werden. Dies verhindert, dass das KI-System einen bestimmten Lösungsweg forciert. Stattdessen ermöglicht das KI-System den Studierenden, einen alternativen Lösungsweg zu finden. Hierfür wurde zunächst ein Prototyp entwickelt, der a priori generierte Trainingsdaten nutzt. Das heißt, dass anhand der Musterlösung zunächst verschiedene Lösungsvarianten und Fehler umgesetzt werden. Für diese Lösungsvarianten und Fehler werden die Zwischenergebnisse und Plots gespeichert. Zusätzlich werden Fehlerklassen definiert, die den Lösungsvarianten und Fehlern zugeordnet werden. Anhand dieser synthetisch erzeugten Trainingsdaten werden künstliche neuronale Netze mittels überwachten Lernens trainiert, um die richtigen und falschen Lösungen zu klassifizieren. Die so trainierten künstlichen neuronalen Netze bilden somit einen essenziellen Bestandteil des KI-Systems.

In Abbildung 2 wird die Funktionsweise des KI-Systems im Kontext des Lehrbetriebes verdeutlicht. Dabei wird davon ausgegangen, dass das KI-System bereits a priori anhand von Trainingsdaten trainiert ist. In den ersten beiden Schritten in Abbildung 2 wird eine Vorlage bereitgestellt, die an geeigneten Stellen durch den Programmcode der Studierenden ergänzt werden soll. Hierdurch wird den Studierenden ein Bearbeitungsrahmen gegeben, der den Fokus auf die jeweiligen Lerninhalte der Übung legt. Bis hierhin entspricht das Übungskonzept den bisherigen Lehrkonzepten in Programmierübungen (grau hinterlegt in Abbildung 2).

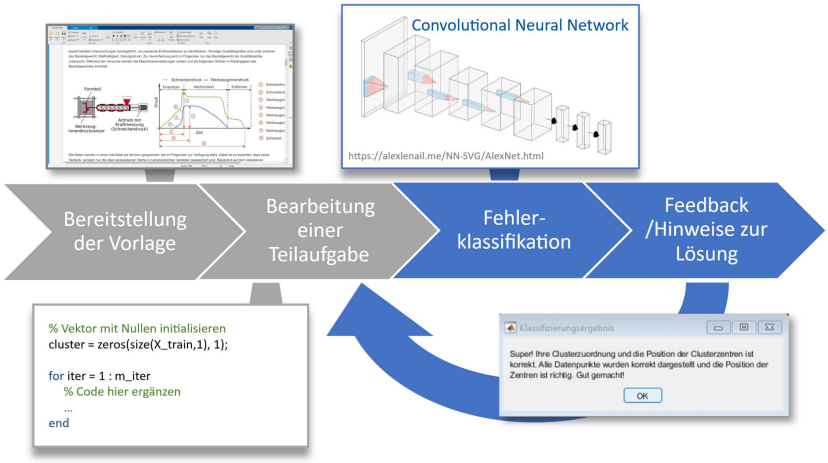


**Abbildung 2:** Funktionsweise des KI-Systems im Lehrbetrieb, unterteilt in vier Schritte

Werden die Teilabschnitte des Programmcodes von den Studierenden ausgeführt, kommt es (verborgen für die Studierenden) zur Fehlerklassifikation. Das heißt, dass die a priori trainierten, künstlichen neuronalen Netze anhand der erzielten Zwischenergebnisse und Plots eine Fehlerklassifikation durchführen. Anschließend gibt das KI-System anhand der Fehlerklassifikation den Studierenden zunächst einen Hinweis, ob die Lösung fehlerfrei oder fehlerhaft ist. Bei wiederholt falscher Zwischenlösung werden die Hinweise des KI-Systems umfangreicher. Die Studierenden erhalten so zunehmend detaillierteres Feedback bzw. Hinweise, um ihren implementierten Code weiterzuentwickeln bzw. zu korrigieren. Hierdurch sollen häufig auftretende Probleme der Studierenden bereits während der Selbstlernphase von den Studierenden gelöst werden können. Die menschlichen Tutor:innen können sich dann während der Präsenzzeit verstärkt auf weitergehende und individuelle Detailfragen der Studierenden konzentrieren.

### 3.3 Technische Umsetzung

Die Programmieraufgaben der oben genannten Lehrveranstaltung erfolgen in MATLAB/Simulink. Für eine möglichst einfache Integrierbarkeit und Kompatibilität ist auch das KI-System in MATLAB/Simulink implementiert. Jedoch ist das KI-System so generisch konzipiert, dass es konzeptionell mit geringem Aufwand auf andere Programmiersprachen und Übungen übertragen werden kann.

Zum gegenwärtigen Zeitpunkt wurde das KI-System als Prototyp entwickelt, der lokal auf den Computern der Studierenden ausführbar ist. Das bedeutet, dass die Studierenden für jede Programmieraufgabe ein MATLAB-Live-Skript und ggf. ein zusätzliches Simulinkmodell erhalten. Das MATLAB-Live-Skript enthält die Aufgabenbeschreibungen sowie ein Grundgerüst für den zu implementierenden Code. Nach geeigneten Teilaufgaben wird das KI-System im Hintergrund aufgerufen, sodass die Zwischenergebnisse der Studierenden klassifiziert werden. Im weiteren Projektverlauf wird das KI-System auf einen Server implementiert, sodass die Zwischenergebnisse anonymisiert an den Server geschickt, von diesem verarbeitet werden und den Studierenden eine geeignete Hilfestellung gegeben wird. Hierdurch erhält das KI-System während des Semesters kontinuierlich neue Trainingsdaten, sodass sich das KI-System kontinuierlich verbessern kann. Darüber hinaus können die anonymisierten Daten zur Analyse der Lernerfolge genutzt werden, sodass sich die Tutor:innen entsprechend auf die Präsenzzeit vorbereiten können.

## 4 Evaluation

Der entwickelte Prototyp des KI-Systems wurde im regulären Lehrbetrieb im Rahmen der ersten Übungseinheit des Semesters eingesetzt. Zur begleitenden Evaluierung wurde im Rahmen der Lehrveranstaltung eine empirische Untersuchung durchgeführt. Ziel war die Untersuchung der Wirksamkeit und Akzeptanz des KI-Systems im Kontext der Programmierausbildung. Der Fokus lag insbesondere auf der Wahrnehmung der Studierenden hinsichtlich der Unterstützungsqualität, der Nützlichkeit der bereitgestellten Rückmeldungen sowie dem Zusammenspiel mit menschlichen Tutoren.

Die Evaluierung erfolgte mittels eines standardisierten Fragebogens, der unter anderem sieben Aussagen zur Nutzung und Wirkung des KI-Systems umfasste. Die Teilnehmenden bewerteten jede Aussage auf einer Likert-Skala von 1 („sehr gut“) bis 5 („sehr schlecht“). Die Ergebnisse sind in Abbildung 3 dargestellt.

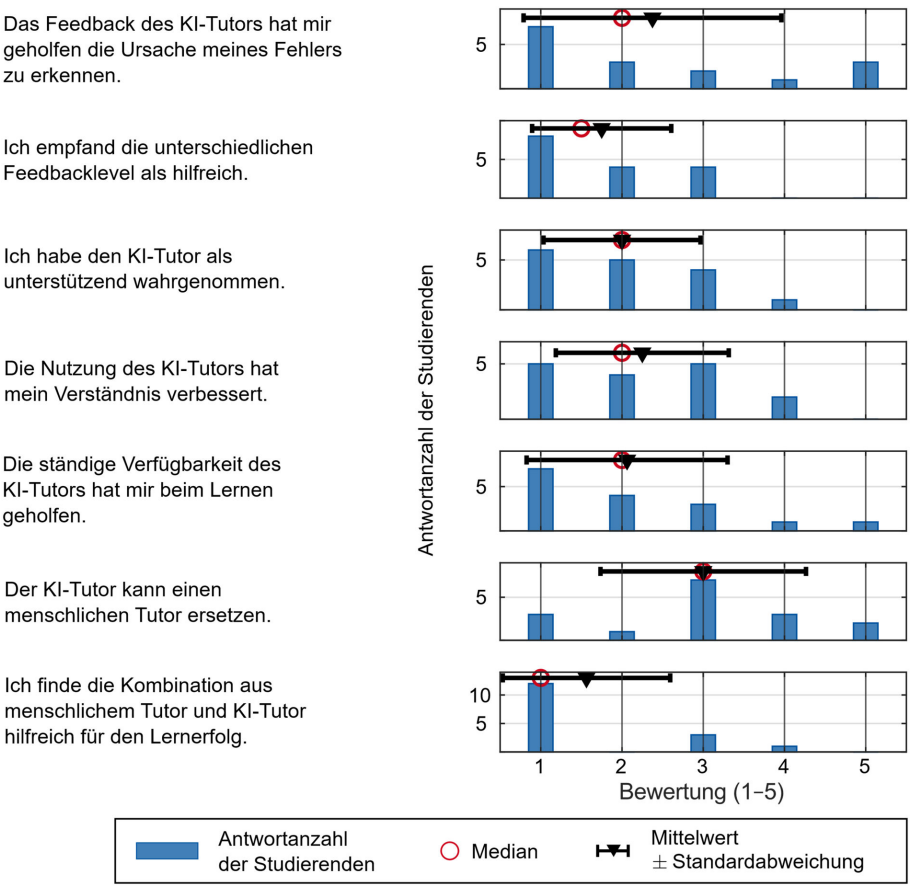


**Abbildung 3:** Ergebnisse einer ersten Befragung von 16 Teilnehmenden des Kurses basierend auf der ersten Übung der Lehrveranstaltung

Die aggregierten Ergebnisse der Evaluierung zeigen ein überwiegend positives Bild hinsichtlich des Mehrwertes eines KI-Systems. So hat das Feedback des KI-Systems einem Großteil der Studierenden dabei geholfen, die aufgetretenen Fehler zu identifizieren. Lediglich drei Studierenden hat das KI-System bei der Fehlersuche nicht geholfen. Auch die unterschiedlich ausführlichen Feedbacklevel wurden als überwiegend hilfreich bewertet. Jedoch wurde das KI-System nicht immer als unterstützend wahrgenommen. Hier gilt es in weiteren Evaluationen die Ursachen zu untersuchen. Es lässt sich vermuten, dass die Ursachen für Fehler und Unsicherheiten an ein und derselben Stelle des Codes sehr vielfältig sein können. So ist die Frage zu stellen, ob beispielsweise Informationen über die individuellen Vorkenntnisse der Studierenden berücksichtigt werden können, um das Feedback des KI-Systems zu individualisieren. Dies zeigt sich auch darin, dass die Verständnisförderung sehr unterschiedlich wahrgenommen wurde.

Etwas positiver wurde die ständige Verfügbarkeit des KI-Systems bewertet, was auf das Potenzial zur Ergänzung asynchroner Lernangebote hinweist. Auf der anderen Seite ist klar festzuhalten, dass das KI-System von den Studierenden nicht als gleichwertiger Ersatz für menschliche Tutor:innen wahrgenommen wird. Vielmehr wird die Kombination aus KI-System und menschlichen Tutor:innen von den Studierenden als hilfreich für den Lernerfolg wahrgenommen. Dies bestärkt das angestrebte Lehrkonzept, bestehend aus Selbstlernphasen mit dem KI-System und einer Präsenzphase mit menschlichen Tutor:innen. Zusammenfassend lässt sich daher festhalten, dass der Prototyp bereits auf überwiegend positive Resonanz stößt. Das vorgeschlagene KI-System bietet somit eine vielversprechende Ergänzung zu klassischen Lehrkonzepten und damit zur Unterstützung des Lernprozesses. Aus diesem Grund wird das KI-System im folgenden Wintersemester auf alle Übungen der Lehrveranstaltung „Maschinelles Lernen in der industriellen Regelungstechnik“ angewendet und erneut evaluiert.

## 5 Diskussion

Alle heute bekannten Ansätze zur Unterstützung von Programmierübungen verfolgen das Ziel, Code automatisch analysieren bzw. bewerten zu können, um den Korrekturaufwand in Form von Prüfungen o. Ä. zu reduzieren. Dabei bieten die Ansätze auch viele Möglichkeiten zur selbstständigen Bearbeitung von Programmieraufgaben. Jedoch haben alle bekannten Ansätze gemeinsam, dass in aller Regel der Code analysiert bzw. der Code mit einer Musterlösung verglichen wird. Das vorgestellte KI-System verfolgt dagegen einen Perspektivwechsel, indem es vor allem Zwischenergebnisse in Form von Zeitverläufen und Plots betrachtet. Hierdurch sollen individuelle Lösungswege gefördert werden. Zudem ist das KI-System jederzeit verfügbar, sodass es vor allem individuelle Selbstlernphasen unterstützt. Das KI-System ermöglicht aber auch

1. kollaborative Selbstlernphasen,
2. Präsenzphasen und
3. hybride Lernphasen.

Hier kann das KI-System beispielsweise Informationen zu häufigen Fehlern sammeln und in kollaborativen Selbstlernphasen Studierende bei der Bildung geeigneter Lerngruppen unterstützen. Mithilfe von Gamification-Elementen sind auch spielerische Ansätze zur Steigerung der Lernmotivation denkbar.

Im Rahmen von Präsenzphasen kann das KI-System häufig auftretende Probleme bereits adressieren oder bei korrekter Lösung zusätzliche Sicherheit bieten, sodass sich die zur Verfügung stehenden Tutor:innen auf die Beantwortung spezifischer Detailfragen konzentrieren können. Tutor:innen sollen durch den Einsatz des KI-Systems keineswegs ersetzt werden, sondern diese bei der Beantwortung von Fragen gezielt unterstützen. Auch können mit dem KI-System hybride Lernphasen realisiert werden, die sich beispielsweise aus einer vorbereitenden Selbstlernphase und einer Präsenzphase zusammensetzen. So können in der Präsenzphase alle Schwierigkeiten mit den Tutor:innen diskutiert werden, die nicht in der Selbstlernphase gelöst werden konnten. Selbst Studierende, die die Übung fehlerfrei bearbeiten, profitieren vom KI-System. So kann ihnen die Rückmeldung über die Richtigkeit ihrer Lösung zusätzliche Sicherheit geben.

Aktuell wird das System in einem Kurs mit geringer Gruppengröße erprobt und entwickelt. Perspektivisch soll das KI-System auch bei Kursen mit hoher Gruppengröße Anwendung finden, bei denen der Mangel an Tutor:innen das Anbieten einer Programmierübung aktuell erschwert bzw. verhindert.

Das vorgestellte KI-System bietet somit vielfältige Einsatzmöglichkeiten in der Lehre. Darüber hinaus ist es konzeptionell auf andere Programmiersprachen bzw. andere Programmierübungen übertragbar. Die größte Herausforderung besteht aktuell noch in der Generierung geeigneter Trainingsdaten. Dabei muss die Datenbank, die die Fehlerfälle enthält, kontinuierlich aktualisiert werden, um auch solche Fehlerfälle zu erfassen, die ursprünglich nicht in der Datenbank enthalten waren. Hier sind in der Zukunft weitere Ansätze und Lehrkonzepte zu untersuchen.

## Anmerkungen

Diese Arbeit entstand im Projekt „KI-unterstützte Programmierübung“ und wurde von der Stiftung Innovation in der Hochschullehre im Rahmen der Ausschreibung „Freiraum 2023“ gefördert. Wir danken der Stiftung vielmals für die Unterstützung.

## Literatur

Ananiadou, K. & Claro, M. (2009). *21st Century Skills and Competences for New Millennium Learners in OECD Countries, OECD Education Working Papers*, 41. OECD Publishing. https://doi.org/10.1787/218525261154

Baker, R. & Siemens, G. (2014). Educational data mining and learning analytics. In R. K. Sawyer (Hrsg.), *The Cambridge handbook of the learning sciences* (S. 253–272). Cambridge University Press. https://doi.org/10.1017/CBO9781139519526.016

Bloom, B. S. (1956). *Taxonomy of educational objectives: The classification of educational goals; Handbook I: Cognitive domain*. David McKay.

de Silva, D., Samarasekara, P. & Hettiarachchi, R. (2023). *A comparative analysis of static and dynamic code analysis techniques*. TechRxiv. https://doi.org/10.36227/techrxiv.22810664.v1

Fonte, D., da Cruz, D., Gançarski, A. L. & Henriques, P. R. (2013). A flexible dynamic system for automatic grading of programming exercises. In J. M. Fernandes, R. L. Aguiar & R. J. Machado (Hrsg.), *Proceedings of the 2nd Symposium on Languages, Applications and Technologies (SLATE 2013), Open Access Series in Informatics (OASIcs),* 29 (S. 129–144). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. https://doi.org/10.4230/OASIcs.SLATE.2013.129

Jackson, D. & Usher, M. (1997). Grading student programs using ASSYST. In *Proceedings of the twenty-eighth SIGCSE technical symposium on computer science education (SIGCSE '97)* (S. 335–339). Association for Computing Machinery. https://doi.org/10.1145/268084.268210

Liu, R., Zhao, J., Xu, B., Perez, C., Zhukovets, Y. & Malan, D. J. (2025). Improving AI in CS50: Leveraging Human Feedback for Better Learning. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education* (Vol. 1, S. 715–721). Association for Computing Machinery. https://doi.org/10.1145/3641554.3701945

Malan, D. J., Sharp, C., van Assema, J., Yu, B. & Zidane, K. (2021). *CS50's GitHub-based tools for teaching and learning*. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (SIGCSE '21, S. 1354). Association for Computing Machinery. https://doi.org/10.1145/3408877.3432499

Ramos, J., Trenas, M. A., Gutiérrez, E. & Romero, S. (2013). E-assessment of Matlab assignments in Moodle: Application to an introductory programming course for engineers. *Computer Applications in Engineering Education, 21*(5), 728–736. https://doi.org/10.1002/cae.20520

Sharma, T., Kechagia, M., Georgiou, S., Tiwari, R., Vats, I., Moazen, H. & Sarro, F. (2021). A survey on machine learning techniques for source code analysis. *arXiv*. https://doi.org/10.48550/arXiv.2110.09610

Striewe, M. & Goedicke, M. (2014). A review of static analysis approaches for programming exercises. In M. Kalz & E. Ras (Hrsg.), *Computer assisted assessment. Research into e-assessment* (S. 100–113). Springer. https://doi.org/10.1007/978-3-319-08657-6_10

Truong, N., Roe, P. & Bancroft, P. (2004). Static analysis of students' Java programs. In *Proceedings of the Sixth Australasian Conference on Computing Education* (ACE '04), Vol. 30 (S. 317–325). Australian Computer Society.

Truong, N., Roe, P. & Bancroft, P. (2005). Automated feedback for "fill in the gap" programming exercises. In D. L. Tolhurst & S. Mann (Hrsg.), *Proceedings of the 7th Australasian Conference on Computing Education* (ACE '05), Vol. 42 (S. 117–126). Australian Computer Society.

Wang, T., Su, X., Ma, P., Wang, Y. & Wang, K. (2011). Ability-training-oriented automated assessment in introductory programming course. *Computers & Education, 56*(1), 220–226. https://doi.org/10.1016/j.compedu.2010.08.003

## Autoren

Dr. Sebastian Stemmler. RWTH Aachen University, Institut für Regelungstechnik, Aachen, Deutschland; Orchid-ID: 0009-0003-2079-3431; E-Mail: s.stemmler@irt.rwth-aachen.de

Jens Ahlers. RWTH Aachen University, Institut für Regelungstechnik, Aachen, Deutschland; E-Mail: j.ahlers@rwth-aachen.de

Robert Göllinger. RWTH Aachen University, Institut für Regelungstechnik, Aachen, Deutschland; E-Mail: r.goellinger@rwth-aachen.de

**Zitiervorschlag**: Stemmler, S., Ahlers, J. & Göllinger, R. (2026). KI-unterstützte Programmierung mittels ergebniszentrierter Fehlerklassifikation. Potenziale zur Schaffung neuer Lernräume. *die hochschullehre*, Jahrgang 12/2026. DOI: 10.3278/HSL2618W. Online unter: wbv.de/die-hochschullehre

# die hochschullehre

## Interdisziplinäre Zeitschrift für Studium und Lehre

Die Open-Access-Zeitschrift **die hochschullehre** ist ein wissenschaftliches Forum für Lehren und Lernen an Hochschulen.

Zielgruppe sind Forscherinnen und Forscher sowie Praktikerinnen und Praktiker in Hochschuldidaktik, Hochschulentwicklung und in angrenzenden Feldern, wie auch Lehrende, die an Forschung zu ihrer eigenen Lehre interessiert sind.

**Themenschwerpunkte**

- Lehr- und Lernumwelt für die Lernprozesse Studierender
- Lehren und Lernen
- Studienstrukturen
- Hochschulentwicklung und Hochschuldidaktik
- Verhältnis von Hochschullehre und ihrer gesellschaftlichen Funktion
- Fragen der Hochschule als Institution
- Fachkulturen
- Mediendidaktische Themen

wbv.de/die-hochschullehre

Alle Beiträge von **die hochschullehre** erscheinen im Open Access!

wbv Publikation · wbv Media GmbH & Co. KG · service@wbv.de · wbv.de

wbv