Individuelle Förderung von Programmierfertigkeiten im Studium am Beispiel von Intelligenten Tutor Systemen für SQL

Anna M. Thaler, Franziska K. Paukner, Jonas-Dario Troles & Ute Schmid

Trotz vermehrter Technologisierung im (Hoch-)Schulkontext finden konkrete auf Künstlicher Intelligenz (KI) basierte Ansätze in den Lehr- und Lernumgebungen relativ wenig Anwendung. In diesem theoriebasierten Konzeptpapier erklären wir Einsatzmöglichkeiten von KI-basierten Methoden in der Hochschullehre anhand von sog. Intelligenten Tutor Systemen (ITS) im Anwendungskontext der Datenbanksprache SQL und allgemeiner Programmierfertigkeiten. Wir zeigen individuelle, technische und organisationale Rahmenbedingungen für eine erfolgreiche Umsetzung von ITS innerhalb der Hochschullehre auf. Es wird als Literaturüberblick dargestellt, wie mit ITS prozedurales und deklaratives Wissen ermittelt und die anzueignenden Fertigkeiten durch individuelles Feedback ausgebaut werden können. Insbesondere gehen wir dabei auf Overlay-Modelle und Algorithmic Debugging ein. Als interaktiver Feedbacktyp werden strukturanaloge Beispiele als Anwendung analogen Schlussfolgerns diskutiert. Als Reflexion des Forschungsprozesses beschäftigen wir uns mit relevanten theoretischen Überlegungen für die akademische Lehrpraxis, insbesondere der Schaffung einer offeneren Fehlerkultur und produktiven Scheiterns sowie kritischer Überlegungen zu Evaluationsmethoden bestehender ITS.

1 Einleitung

Der Erwerb von Programmierfertigkeiten stellt eine kognitiv komplexe Aufgabe mit vielen Freiheitsgraden dar. In der natürlichen Sprache kann ein und derselbe Sachverhalt mit unterschiedlichen Wörtern und Satzstrukturen umgesetzt werden, ohne dass sich die Bedeutungsebene verändert. Gleiches gilt auch für Programmier-Aufgaben, die auf verschiedene Arten korrekt in Code umgesetzt werden können. Dies ist nicht zu vergleichen mit einer Mathematik-Textaufgabe, die ebenso als natürlichsprachige Anforderung formuliert sein kann, aber letztendlich auf eine vorher festgelegte Lösung in Form von Gleichungen abzielt. Lernen zu programmieren benötigt neben konzeptuellem Wissen bezüglich der Syntax und Semantik einer Programmiersprache überwiegend prozedurales Wissen und Fertigkeiten, welche in einer komplexen Problemlöse-Umgebung gelernt werden müssen (Anderson et al., 1989). Daher ist reines Auswendiglernen eines bereits funktionierenden Codes für ein gegebenes Problem unzureichend und wird zu keinem tiefen Verständnis der Programmiersprache führen (Shah et al., 2017).

Das Aneignen einer Programmiersprache fördert daher allgemeines problemlöseorientiertes Denken und ist von der Schule bis hin zur Erwachsenenbildung eine wichtige Fertigkeit, um Wissen auf neue Problemstellungen transferieren zu können, weshalb sog. *Computational Thinking* bereits in einigen

Ländern in das Curriculum für Mathematik oder digitale Kompetenzen ergänzt wurde (Kilhamn et al., 2021; Moreno-León et al., 2019). Programmieren ist zudem kein Thema, das sich ausschließlich auf ein Informatik-Studium beschränkt.

Ein Beispiel dafür ist die Datenbanksprache SQL – kurz für Structured Query Language. SQL ist eine deklarative Datenbanksprache, welche es ermöglicht, Datenbankkomponenten zu erstellen, darauf mithilfe von Anfragen zuzugreifen, Daten zu manipulieren und Zugriffsrechte zu kontrollieren. Da eine Datenbank nahezu jede Information abbilden kann – ob Daten von Mitarbeiter*innen einer Firma oder Medien einer Bibliothek – bleibt sie domänenunspezifisch und findet daher breite Anwendung in weiteren Fachgebieten. Für eine fundierte Auseinandersetzung mit einer Programmiersprache muss diese allerdings in diversen Aufgabenstellungen durch eigenes Ausprobieren angeeignet werden. Fehler sind hierbei ein unvermeidliches Nebenprodukt während des Lernprozesses und können zugleich neue Lernmöglichkeiten darstellen (VanLehn, 1998). So werden Studierende beim ersten Versuch, das Problem des Turms von Hanoi in der Programmiersprache Python umzusetzen, nicht ohne Fehlermeldungen auskommen. Im Gegensatz zu vielen anderen Disziplinen, sind diese Fehlermeldungen in der Informatik sogar erwünscht. Sie ermöglichen, die eigenen Fehler zu identifizieren, sie zu beheben und daraus zu lernen (Shah et al., 2017).

Leider wird im Laufe der Schullaufbahn, aber auch in der Hochschullehre oft darauf abgezielt, den Fehleranteil möglichst gering zu halten. Bei Abschlussprüfungen werden zum einen falsche Antworten mit Punkteabzug ‚bestraft'. Dies kann je nach interindividuell unterschiedlicher Veranlagung und maladaptiven Attributionsmustern zu reduzierter Anstrengung, negativen Emotionen und geringerer Leistung führen (Diener & Dweck, 1978). Zum anderen wird das behandelte Thema nach Bestehen der Prüfung von den Lernenden abgehakt und es entsteht keine neue Lernmöglichkeit, um aus den Fehlern neue Erkenntnisse zu generieren. Für eine korrekte und effiziente Fehleridentifikation benötigt es daher auch metakognitive Strategien zum selbstgesteuerten Lernen und zur Emotionsregulierung (Darabi et al., 2018, S. 1102), welche gezielt vermittelt oder angeregt werden sollten.

Um die (Hochschul-)Lehre, stärker hinsichtlich individueller Voraussetzungen und Bedürfnisse zu optimieren, können hierfür zwei grundlegende KI-basierte Ansätze verwendet werden. Zum einen können datengetriebene Ansätze wie das sog. Learning Analytics verwendet werden, welches Vorhersagen auf Basis von dynamisch generierten Daten der Studierenden trifft (Koedinger et al., 2013). Hierbei werden bspw. Lernniveaus zugeteilt und auf Grundlage der Performanz der Lernenden Empfehlungen für neue Lernaufgaben gegeben. Dabei werden häufig datenintensive Machine Learning-Algorithmen und weitere Ansätze der KI als sog. Black Box verwendet, bei denen Anwendende in deren internale Vorgänge keinen Einblick bekommen. Das könnte eine Beurteilung erschweren, ob die ausgegebenen Daten tatsächlich auf dem internalen Zustand der Lernenden basieren. Aufgrund des behavioralen Charakters kann diese Methode zwar sehr generisch für unterschiedliche Lehrinhalte verwendet werden, allerdings scheint dieser Fokus auf reine Verhaltensmaße als Entscheidungsbasis den kognitiven Anteil des Lernprozesses zu vernachlässigen.

Learning Analytics kann ein informatives Werkzeug zur Unterstützung der Lehrkräfte sein. Da jedoch die Lernenden und deren internaler Wissensstand in den Evaluationsprozess nicht direkt mit einfließen, eignet es sich unserer Ansicht nach nicht als hinreichende Diagnose des tatsächlichen Wissensstandes und Lernfortschritts der Lernenden. Anders wird dies bei der zweiten Möglichkeit zur In-

tegration von KI in der Hochschullehre umgesetzt, den sog. ITS. Diese erstellen ein explizites Modell, das sog. Lernenden-Modell, das tatsächliche sowie ggf. missverstandene Konzepte und nicht nur das auf behavioraler Ebene gezeigte Wissen der Lernenden abbildet. Das Lernenden-Modell kann mit dem korrekten Wissen der jeweiligen Domäne verglichen werden, um bei Abweichungen intelligentes Feedback auf individueller Ebene zu geben, und ermöglicht so konstruktive handlungsorientierte Ansätze des Lehrens und Lernens. Ein ITS als Ergänzung in der Hochschullehre ermöglicht zum einen eine direkte Feedback-Schleife für die Gestaltung der Lehre, denn die Lehrkräfte erhalten Rückmeldung über das Ausmaß der erfolgreich vermittelten Lehrinhalte und noch mangelhaft verstandenen Konzepten. Außerdem sind ITS an sich ein Forschungsgegenstand, sodass mithilfe der Daten der Lernenden determinierende Faktoren interindividueller Unterschiede in Lösungsstrategien von z. B. Novizen und Fortgeschrittenen besser erforscht werden können. ITS bieten daher eine Möglichkeit, Forschung direkt in angewandte Lehr- und Lernumgebungen mit einzubinden und diese stärker miteinander zu vernetzen.

Als Methode des Scholarship of Teaching and Learning-Ansatzes führen wir im Folgenden ein ausführliches Literature Review durch (MacMillan, 2018), in dem wir den aktuellen Forschungsprozess zur Entwicklung und Anwendung von ITS in der Lehrpraxis kritisch reflektieren und eine theoretische Basis für relevante Fragen für die Lehrpraxis schaffen. Dabei gehen wir zunächst in Kapitel 2 auf unsere Idee einer offeneren Fehlerkultur ein und zeigen Vorteile von analogem Schlussfolgern im Kontext von ITS. Basierend auf der theoretischen Erarbeitung dieses Konzeptpapiers, arbeiten wir an einem ITS für die Programmiersprache SQL, welche den Kritikpunkten bisheriger Systeme entgegenwirken soll. Konkret sollen mehrere Methoden zu einem kohärenten System kombiniert werden, bspw. ein Overlay-Modell mit einer Fehlerbibliothek und die Möglichkeit für einen Sokratischen Dialog.

Für ein besseres Verständnis dieser Begriffe werden wir die Grundideen von ITS und die unterschiedlichen Gestaltungsmöglichkeiten in Kapitel 3.1 und 3.2 genauer erläutern und kritisch hinterfragen. In Kapitel 3.3 gehen wir insbesondere kritisch auf bisherige Evaluationsmethoden ein und zeigen Lücken in der Standardisierung bestehender Methoden auf. Dies soll ebenfalls als Basis dienen, um darauf aufbauend Leitfäden für die Empirie zu entwickeln, die es ermöglichen, unterschiedliche ITS in der Lehrpraxis zu evaluieren und deren Effektivität für den Lernprozess zu beurteilen. Im vierten Kapitel gehen wir auf individuelle, technische und organisationale Rahmenbedingungen für eine erfolgreiche Umsetzung von ITS innerhalb der Hochschullehre ein, und fassen unseren Beitrag im fünften Kapitel mit anschließender Diskussion zusammen.

Unser Forschungsvorhaben zielt darauf ab, die Vielfalt guter menschlicher Pädagog*innen in ein ITS einzubauen, um diese somit ideal im Lehralltag zu unterstützen. Als Forschungsfragen leiten uns dabei folgende Fragen: Welche ITS-Methode ist für welches Fach besser geeignet als andere? Wie unterscheidet sich der Lernfortschritt in Transferaufgaben nach Lernen mit herkömmlichen E-Learning-Plattformen im Vergleich zu einem ITS? Sind E-Learning-Plattformen als Ergänzung des Unterrichts für vermeintliche ‚Auswendig-Lern-Fächer' in gleichem Maße geeignet wie ein ITS, das individualisiertes Feedback gibt?

Wir sind der Auffassung, dass auch diese ‚Auswendig-Lern-Fächer' auf einem grundlegenden Verständnis von Zusammenhängen basieren, und möchten dieser Annahme mithilfe der Entwicklung eines ITS und dessen Evaluierung in der Lehrpraxis mit geeigneten Empirie-Methoden nachgehen.

2 Analogien als Lernmöglichkeit im Umgang mit Fehlern als produktives Scheitern hin zu einer offeneren Fehlerkultur

2.1 Problembasiertes Lernen im Umgang mit Fehlern auf konstruktive Art

Zur Förderung von allgemeinen Problemlösestrategien und kritischem Denken können problembasierte Lernumgebungen verwendet werden. Diese zeigen für den Wissensabruf aus dem Langzeitgedächtnis in performanz- oder fertigkeitsbasierten Maßen und beim Abruf einer Kombination aus konzeptuellem Wissen mit prozeduralen Fertigkeiten deutliche Vorteile im Vergleich zu vorlesungsbasierten Lernumgebungen (Dochy et al., 2003; Yew & Goh, 2016). Da diese Konzepte auch in der Programmierung Anwendung finden, scheint dieser Ansatz für die Vermittlung von allgemeinen Programmierkenntnissen relevant zu sein.

Wenn das Erlernen einer Programmiersprache darauf basiert, Fehler zu begehen und diese als Möglichkeit zum neuen Erkenntnisgewinn zu sehen, kann auch von *produktivem Scheitern* (Kapur, 2016) gesprochen werden; das heißt, kurzfristige Fehler führen zu einem langfristigen Lernfortschritt. Hierbei können Fehler als Informationsquelle und wertvolle Rückmeldung dienen, müssen von den Lernenden aber auch als solche erkannt werden (Frese & Altmann, 1989). So kann beim Programmieren nicht das gesamte Programm verworfen werden, sobald die erste Fehlermeldung erscheint. Hier wird nicht nur Geduld trainiert, sondern auch die Erkenntnis angeregt, dass eigene Schwachstellen mithilfe von Fehlern leichter identifiziert werden können. Sobald ein Programm funktioniert, wird deutlich, dass Fehler generell behebbar sind und somit nicht pauschal negativ attribuiert werden müssen (Diener & Dweck, 1978). Typischerweise wird den Lernenden beim produktiven Scheitern die Möglichkeit gegeben, selbstständig ein gegebenes (komplexes) Problem zu lösen, und erst bei Schwierigkeiten – seien es fehlerhafte Lösungen oder unvollständige Antworten – entsprechende Instruktionen als Feedback oder Hilfestellung gegeben (Kapur, 2008). Fehler mit Feedback führen nicht nur zu schnellerer und besserer Performanz in darauffolgenden Aufgaben im Vergleich zum gleichen Zeitaufwand ohne Fehler (Kornell et al., 2009, S. 995), sondern zeigen auch positive Effekte auf die Selbstwirksamkeit der Lernenden (Lorenzet et al., 2005, S. 315) und auf den Abruf der Informationen aus dem Langzeitgedächtnis (Kornell et al., 2009, S. 996).

Ob die Möglichkeit, aus Fehlern neues Wissen zu generieren, tatsächlich genutzt wird, hängt von mehreren Faktoren ab. Zum einen benötigen die Lernenden generelle metakognitive Fähigkeiten wie selbstgesteuertes Lernen (Darabi et al., 2018, S. 1102), zum anderen hängt das Lernen aus Fehlern von der interindividuell unterschiedlichen Fähigkeit zur Emotionsregulierung, motivationalen Prozessen (Tulis et al., 2016) sowie den während des Fehlers vorherrschenden temporären oder stabil überdauernden Emotionen ab (Steuer, 2014, S. 40). Lishinski et al. (2016) konnten zeigen, dass sich die Zusammenhänge zwischen selbstregulierenden Fähigkeiten wie Selbstwirksamkeit, Zielorientierung oder metakognitive Strategien und der Erfolgsprädiktion von Lernenden bei einführenden Programmieraufgaben geschlechterspezifisch unterscheiden. Tulis et al. (2016) haben daher einen

umfassenden theoretischen Rahmen entworfen, der den fehlerbasierten Lernprozess bezogen auf motivationale Änderungen, Selbstregulierungsprozesse, metakognitive Aktivitäten und emotionale Erfahrungen einordnet. Produktives Scheitern ist daher stark individuen- und kontextabhängig und verdeutlicht, dass diese Faktoren im Lernprozess, z. B. mithilfe eines ITS, das auch emotionale und motivationale Zustände modellieren kann, ergänzend berücksichtigt werden sollten. Diese interindividuellen Unterschiede können mithilfe eines ITS, das Daten aus der Lehrpraxis zieht, gezielt untersucht werden und stellen eine wichtige Basis für die generelle Lernforschung bezogen auf konkrete Anwendungsfächer wie z. B. der Informatik dar.

2.2 Analogien als nicht-direktives Feedback-Instrument für produktives Scheitern

Da Fehler als Informationsfunktion dienen können (Frese & Altmann, 1989), sollte entsprechendes Feedback als Reaktion auf fehlerhafte Antworten gegeben werden, das dabei unterstützt, Strategien zum Wissenstransfer zu entwickeln (Steenhof et al., 2020, S. 1-2). Dieses Feedback kann diverse Formen annehmen: vom simplen Feedback als ein Aufzeigen, ob die gegebene Lösung korrekt oder fehlerhaft ist, über ein Markieren von fehlerhaften Passagen bis hin zum Vorlegen einer vollständig korrekten Lösung (Mitrovic, 2003). Eine Möglichkeit für ein weniger direktives Feedback bei fehlerhaften Antworten ist das Bereitstellen von strukturanalogen (isomorphen) Beispielen, also bereits gelöste Probleme, die eine gleiche oder ähnliche zugrundeliegende Problemstruktur aufweisen, sich aber in ihrer Oberflächenstruktur unterscheiden (Renkl et al., 1998; Sweller & Cooper, 1985). Wenn bspw. eine Anfrage an eine Datenbank gestellt werden soll, die ausgibt, welche Mitarbeitenden in den letzten fünf Monaten keinen Urlaub beantragt haben, kann als strukturanaloges Beispiel einer anderen Domäne gezeigt werden, wie das Problem für eine Datenbank zu Ferienwohnungen, die in den letzten drei Monaten mindestens Vier-Sterne-Bewertungen erhalten haben, gelöst wird. Das strukturanaloge Beispiel besteht dabei aus der Spezifikation des Problems, sämtlichen Lösungsschritten und der finalen Lösung selber (Renkl et al., 1998). Es wird argumentiert, dass diese Beispiele die kognitive Beanspruchung reduzieren (Sweller, 1994, S. 308).

Dies könnte eine Erklärung für die kürzere Bearbeitungszeit für zukünftige Probleme sein (Chen, 2018). Die isomorphe Eigenschaft eines Beispiel-Problems als Hilfestellung für die Lösung eines gegebenen Problems ist Voraussetzung dafür, dass Prinzipien des analogen Schlussfolgerns angewandt werden können. Die Elementmengen der beiden zu betrachtenden Probleme können verschieden sein, ihre Beziehungen innerhalb des gelösten Problems müssen sich in paralleler Weise zu jenen des zu lösenden Problems verhalten (Gentner & Maravilla, 2017, S. 186). Das strukturanaloge, gelöste Beispielproblem dient hierbei als Quellbereich (Vorlage), dessen Erkenntnisse für die Lösung des Zielbereichs (aktuelles Problem) genutzt werden können. Zunächst muss dafür die zugrundeliegende Struktur extrahiert und in Form von abstrakten Lösungsschemata repräsentiert werden, welche Lernende befähigen, mithilfe eines strukturellen Abgleiches (Gentner & Markman, 1997) die Erkenntnisse der bekannten Domäne (gelöstes isomorphes Problem) auf neue Probleme mit neuen Oberflächencharakteristika korrekt zu transferieren. Da die Gesamtlogik des neuen Problems abweichen kann, muss entsprechend von irrelevanten Attributen und Relationen abstrahiert werden (Renkl et al., 1998).

Es konnte gezeigt werden, dass Nutzer*innen in Problemlöse-Situationen, u. a. bei Programmieraufgaben, Beispiele als Hilfestellung gegenüber textuellen Informationen bevorzugen (Recker & Pirolli, 1995; Renkl et al., 1998). Bei Programmieraufgaben zeigte sich diese Präferenz lediglich bei Novizen, welche analoges Schließen aus isomorphen Beispielen häufig beim ersten Versuch, rekursive Funktionen zu schreiben, anwenden (Pirolli & Anderson, 1985). Fortgeschrittene Studierende fokussieren sich hingegen stärker auf zusätzliche Informationen wie Datenbankschemata bei SQL-Problemen (Najar et al., 2014). Hier wird bereits deutlich, dass die Verwendung und damit der Nutzen von isomorphen Beispielen auch vom Vorwissen der Lernenden abhängt und somit individuell angepasst im Lernprozess angeboten werden sollte. Strukturanaloge Beispiele sind besonders für Systeme zur Vermittlung von prozeduralen Fertigkeiten relevant, z. B. in Domänen wie Musik, Schach, Grundlagen der Mathematik oder Programmierung (Atkinson et al., 2000, S. 185).

Für eine automatisierte Generierung von strukturanalogen Beispielen können problembasierte Mustervorlagen (engl. *problem templates*) für die Kategorisierung der Probleme verwendet werden (Mathews & Mitrovic, 2007). Wir argumentieren, dass analoge Beispiele als Rückmeldung auf fehlerhafte Antworten die Möglichkeit bieten, Wissen zu transferieren, und mithilfe von Generalisierungsprozessen – also der Abstraktion von Schemata (Gentner & Maravilla, 2017) – eine Lerngelegenheit entsteht, die nicht auf dem expliziten Aufzeigen von einzelnen Fehlern beruht und weniger zu einem Fehlerklima beiträgt, das maladaptive Attributionsmuster (Diener & Dweck, 1978) begünstigen könnte. Stattdessen stellt sie ein nicht-direktives Feedback-Instrument dar, das sich positiv auf produktives Lernen auswirken kann.

Die Einbindung von individuell angepassten Feedback-Möglichkeiten wie strukturanalogen Beispielen in die (Hochschul-)Lehre möchten wir generell mithilfe der Entwicklung von ITS begünstigen. Anhand unserer Erfahrungen aus dem Entwicklungsprozess eines ITS für die Datenbanksprache SQL und gegebener Literatur reflektieren wir über die Herausforderungen aus dem Forschungsprozess die von allgemeinen Entwicklungsfragen bis hin zum fertigen System aufkommen.

3 Reflexionen aus dem Forschungsprozess zur Erstellung von ITS für die Datenbanksprache SQL

ITS sind ähnlich zu Desiderata einer guten Lehrkraft aufgebaut: Sie beinhalten korrektes und vollständiges Wissen über den Lehrinhalt (*Domänen-Modell*), verfügen über pädagogische und didaktische Prinzipien zur Vermittlung dessen (*Tutorielles Modell*), kennen typische Missverständnisse und Methoden, wie diesen entgegengewirkt werden kann, und behalten die individuelle Lerngeschichte der Lernenden im Blick (*Lernenden-Modell*) (Nwana, 1990; Meier, 2002). Diese drei miteinander interagierenden Komponenten stellen gemeinsam mit der Benutzer*innenschnittstelle als Kommunikationsmedium zu den Lernenden ein KI-basiertes Tutor-System dar. Dieses zeichnet sich dadurch als *intelligent* aus, dass es Problemstellungen generieren, diese eigenständig lösen und Lösungsstrategien den Lernenden erklären kann (Burns & Capps, 1988). Die Anwendung von KI-Methoden innerhalb der Systeme ermöglicht eine reiche, interaktive und flexibel angepasste Interaktion mit den Lernenden. Ansätze zum natürlichen Sprachverständnis, sowie zur erklärbaren KI scheinen deshalb vielversprechend für den weiteren Fortschritt der ITS-Forschung. Im Folgenden werden zu berück-

sichtigende Aspekte aus dem Forschungsprozess zur Erstellung eines ITS vorgestellt. Speziell gehen wir dabei auf ein ITS zur Vermittlung der Datenbanksprache SQL ein.

3.1 Genereller Aufbau und Funktionsweise eines ITS

Das *Domänen-Modell* beinhaltet die vom System behandelte Wissensdomäne SQL. Es ist in der Lage konzeptuelles SQL-Wissen und problembezogene Aufgabenstellungen, welche den Lernenden präsentiert werden, sowie die Lösungen und Erklärungen dazu direkt im Programm zu generieren, und passt den dargestellten Inhalt individualisiert an den Lernprozess an (Nwana 1990, S. 258). Das *Lernenden-Modell* bildet die Kenntnisse über das Lernverhalten und die Fertigkeiten der Lernenden ab, beinhaltet aber auch Lernenden-Charakteristika, Antworthistorie, generelle Präferenzen und Performanzwerte. Dabei können, durch die Analyse des Verhaltens der Lernenden bei der Problemlösung der Aufgabenstellung Rückschlüsse über das aktuell angeeignete, fehlende oder auch fehlerhafte Wissen gezogen werden (Nwana, 1990, S. 261). Das *Tutorielle Modell* ist die Komponente des ITS, welches die Interaktion mit den Lernenden gestaltet. Die Art dieser Interaktion hängt von der Lehrstrategie ab und kann je nach gewählten pädagogischen Prinzipien zur allgemeinen Gestaltung der Lernsituation und didaktischen Methoden des konkreten Inhaltsbereichs Formen wie Hilfestellungen, Erklärungen oder Generierung weiterer an den Wissensstand angepassten Aufgaben annehmen (Meier, 2002). Diese Eigenschaft des intelligenten Feedbacks je nach Fehler und Fehlertyp mithilfe von konstruktiver und situativer Instruktion auf individueller Ebene der Lernenden unterscheidet ITS von den meisten E-Learning-Plattformen oder anderen Methoden des computerunterstützten Unterrichts (Chen, 2018). ITS-Verhalten kann generell in äußere Schleifen zur Generierung einer vielfältigen Menge an Aufgaben und einer intelligenten Wahl dieser sowie innere Schleifen für Rückmeldungen und Hilfestellungen innerhalb einzelner Schritte der Problemlöse-Aufgabe eingeteilt werden (VanLehn, 2006).

Da der interne kognitive Zustand der Lernenden nicht direkt beobachtbar ist, muss er mithilfe eines Vergleichs des modellierten Fachwissens und der vom System generierten Lösungen mit jenen der Lernenden – bezogen auf ein gegebenes Problem – deduziert werden, um das Lernenden-Modell zu modellieren und stetig zu aktualisieren (VanLehn, 1988). Somit kann fehlendes Wissen und fehlerhaftes Wissen identifiziert und mithilfe geeigneter Lehrstrategien des *Tutoriellen Modells* gezielt angesprochen und behoben werden. Ein Beispiel für solche Lehrstrategien zur Vermittlung von Problemlösestrategien, wie etwa bei einem SQL-Problem benötigt, stellen die bereits erläuterten strukturanalogen Beispiele dar. In alternierender Darbietung mit reinen Problemlöse-Aufgaben erwiesen sie sich als effektive Lehrstrategie (Chen et al., 2020).

3.2 Herausforderungen bei der Umsetzung der einzelnen Module

Auf kurzfristige Sicht ist das Ziel von ITS eine optimale Unterstützung der Lernprozesse entsprechend dem aktuellen Forschungsstand. Langfristig zielen ITS zusätzlich darauf ab, fundamentale Forschungsfragen zu klären, um wissensbasierte Systeme effektiver und robuster zu gestalten (Burns & Capps, 1988). In der intelligenten Lehrumgebung wird auf konstruktives Lernen, konzeptionelle Wissensvermittlung statt Auswendiglernen und ein Gefühl des *self-monitoring* bei Lernenden gesetzt (Burton, 1988). Da es sich bei ITS um ein domänenspezifisches System handelt, braucht es neben den Entwickler*innen mit Informatikhintergrund, Didaktiker*innen und Grafiker*innen für

eine intuitive Verwendung und Ansprechbarkeit des Systems auch Fachvertreter*innen der Domäne. In unserem Beispiel für den SQL-Tutor muss also eine enge Zusammenarbeit mit Forschenden und Praktiker*innen aus dem Bereich Datenbanksysteme bestehen. Die Schwierigkeit liegt darin, die einzelnen Module, bei der zahlreiche Expertisen benötigt werden (Domäne, Diagnose, Instruktion, Mensch-Computer Interaktion, Implementation, Evaluation etc.), über den gesamten Entwicklungsprozess vom Entwurf über die Entwicklung und Anwendung hin zur Evaluation in ein einziges kongruentes System zu integrieren (Burns & Capps, 1988).

Für die drei Hauptmodule werden je nach Domäne und theoretischem Hintergrund der Entwickler*innen unterschiedliche Methoden für die prototypische Umsetzung einzelner ITS verwendet, welche im Folgenden genauer erklärt werden.

Für das *Domänen-Modell* (engl. *expert module*) gibt es nach Anderson (1988) drei Grundansätze: (1) ein Black-Box-Modell, bei dem die internalen Vorgänge nicht verfügbar oder unbrauchbar für Instruktionen sind z. B. SOPHIE (Brown et al., 1975), (2) ein Glass-Box-Modell, häufig basierend auf Wenn-Dann-Regeln, z. B. GUIDON (Clancey, 1986), und (3) ein kognitiver Modellierungs-Ansatz, bei dem versucht wird, eine realistische Simulierung menschlichen Problemlöseverhaltens zu entwickeln, welches starke Relevanz für die Grundlagenforschung der Kognitionswissenschaft hat. Allgemein bietet es sich an, bestehende Expertensysteme um die Komponenten des Lernenden- und Tutoriellen Modells zu erweitern, um bereits aufgebrachte Entwicklungsanstrengungen und Expertise der Domäne optimal für ITS zu nutzen (Anderson, 1988).

Im Domänen- sowie im Lernenden-Modell muss berücksichtigt werden, welche Wissensarten geprüft und wie diese in den Modellen repräsentiert werden. Es wird unterschieden zwischen deklarativem Wissen – verbalisierbarem Wissen über Fakten und Konzepte sowie deren Beziehungen zueinander – und prozeduralem Wissen, bei dem das Domänenwissen direkt in Prozesse für Anwendungen spezifischer Fertigkeiten, wie etwa der Fertigkeit, eine korrekte Anfrage in SQL zu formulieren, eingebettet ist (Anderson, 1982). Im Domänenmodell kann deklaratives Wissen beispielsweise mithilfe eines *mixed initiative*-Dialogs vermittelt werden, indem die Interaktion mit dem System als sokratischer Dialog stattfindet (Anderson, 1988), wie es z. B. im ITS zur Geografie Südamerikas namens SCHOLAR (Carbonell, 1970) umgesetzt wird. Hierbei werden sowohl Instruktionen als auch Wissensabfragen mithilfe von Fragen gehandhabt.

Für die Modellierung konzeptionellen Wissens bietet sich als *Lernenden-Modell* ein sog. *Overlay-Modell* an, welches nichtgezeigtes Wissen der Lernenden – häufig in einem Wissensgraf (semantischen Netz) – entsprechend als fehlend markiert. Für einen einfacheren Abgleich wird dabei typischerweise die gleiche Wissensrepräsentation für das Domänen-Modell als geteilte Wissensbasis gewählt, sodass das Wissen der Lernenden eine (echte) Teilmenge des Domänen-Wissens darstellt (VanLehn, 1988). Der Nachteil dieses Ansatzes ist, dass Missverständnisse im Lernenden-Modell nicht abgebildet werden können.

Für die Abfrage von prozeduralem Wissen, also dem Zeigen von Fertigkeiten in problembasierten Aufgabenstellungen, kann eine Fehlerbibliothek über typische missverstandene Konzepte (Burton, 1982) verwendet werden.

Diese hat allerdings einen Korrektheits- und Vollständigkeitsanspruch und muss mit größerem Aufwand dynamisch während des Lernprozesses mithilfe von strukturierten Interviews oder aus der Literatur erzeugt werden (VanLehn, 1988).

Eine weitere Möglichkeit für die Umsetzung des Lernenden-Modells besteht daher darin, verstandenes und missverstandenes (prozedurales) Wissen nicht explizit zu sammeln, sondern mithilfe sog. *Constraints* zu erschließen. Constraints sind eine Menge abstrakter (Teil-)Regeln, die für eine korrekte Umsetzung eines gegebenen Problems, z. B. bei schriftlichem Subtrahieren, benötigt werden. Sobald Lernende einen oder mehrere dieser Constraints verletzten, können fehlerhaftes Wissen oder Missverständnisse dynamisch im Lernprozess identifiziert und somit gezielt angesprochen werden (Ohlsson, 1992). Diese Methode findet Anwendung im SQL-Tutor von Mitrovic (2003).

Ein Ansatz, der ebenfalls den Vorteil hat, dass keine missverstandenen Konzepte der Lernenden modelliert werden müssen, und gleichzeitig einen natürlich-sprachigen Dialog zwischen Lernenden und dem lehrenden System ermöglicht, ist das *Algorithmic Debugging* für ITS (Zeller & Schmid, 2017; Zinn, 2013). Hierbei wird die Lösung der Lernenden zunächst als korrekt betrachtet, um dann algorithmisch zu ermitteln, von welcher korrekten Lösungsregel sich die Teilschritte unterscheiden, um somit auf fehlerhafte Konzepte rückzuschließen. Zeller und Schmid (2017) stellen in Anwendung dieser Methode einen Ansatz für ITS zum schriftlichen Subtrahieren vor, der das korrekte Vorgehen in Form von Regeln in Prolog abbildet und mithilfe von automatisch generierten isomorphen Beispielen identifizierte Missverständnisse hervorhebt. Eine prototypische Umsetzung dieser Methode befindet sich im ITS namens *Subkraki* (Giacomazzi et al., 2021) der Universität Bamberg.

Der diagnostische Schluss auf das Wissen der Lernenden kann sich nach VanLehn (1988) in seiner Komplexität unterscheiden. Abgesehen vom Typ des Zielwissens hängt der Aufwand des Lernenden-Modells maßgeblich von der Bandbreite der Eingangsinformationen für das ITS ab. Diese kann variieren zwischen der Abbildung von reinen Endzuständen, z. B. im Tutor PROUST für die Programmiersprache LISP (Johnson & Soloway, 1985), über Zwischenzustände, z. B. in Spade (Miller, 1979), bis hin zu mentalen Zuständen, z. B. im LISP Tutor (Anderson & Reiser, 1985).

Herausforderungen im diagnostischen Schluss bestehen je nach verwendeten Eingangsinformationen also darin, dass das gezeigte Verhalten in den Aufgaben oder Zwischenschritten direkt auf die mentalen tatsächlichen Zustände der Lernenden zurückzuführen ist. Womöglich könnte eine Kombination aus mehreren Missverständnissen dennoch zur richtigen Lösung führen und je nach Sensibilität des Systems Auswirkungen auf den weiteren Lernprozess haben. Ebenso verhält es sich mit dem Erkennen von Konzentrationsschwächen und unsystematischen Fehlern (Polson et al., 1988).

Für die Erstellung des *Tutoriellen Modelles* muss zusätzlich berücksichtigt werden, um welche Aufgabenstellungen es sich handelt: Werden überwiegend prozedurale Fertigkeiten, wie bspw. in der Formulierung von SQL-Abfragen, verlangt? Müssen Lernende konzeptionelles Wissen abrufen? Oder sollen fehlerhafte Lösungen analysiert, erklärt und ‚debuggt' (Burns & Capps, 1988) werden? Die Aufgabenstellung hat somit Einfluss auf die möglichen Hilfestellungen, die das ITS geben kann. Hier muss wiederum entschieden werden, ob Hilfestellungen ungefragt präsentiert werden, diese auf Verlangen der Lernenden abgerufen werden sollen, welche Art von Feedback mit welchem Ausmaß an Hilfestellung gegeben wird, in welchem Grad die Generation von neuen Problemen auf die Per-

formanz der vergangenen Probleme eingeht usw. (Halff, 1988). Hier soll angemerkt werden, dass nicht in allen ITS die vier Modelle gleichermaßen ausgeprägt sind und teilweise das tutorielle Modell rein aus Heuristiken besteht und somit stark vernachlässigt wird (Conati, 2009).

Bei der Entwicklung der Benutzerschnittstelle muss berücksichtigt werden, dass nicht nur das Thema an sich gelernt werden muss, sondern auch der Umgang mit der Systemtechnik. Durch eine intuitiv verständliche grafische Benutzeroberfläche soll deshalb wenig kognitive Kapazität für das Lernen der Lehrumgebung aufgewandt werden müssen (Miller, 1988). Es wurde angemerkt, dass in der Entwicklung von ITS keine synergetischen Nachhaltigkeitseffekte entstehen können, da die Module teilweise separat entwickelt werden und Leitfäden fehlen, um von bisherigen Arbeiten bestehender ITS zu profitieren (Rodrigues et al., 2005). Dies ist ein wichtiger Ansatzpunkt, der in zukünftiger Forschung stärker berücksichtigt werden sollte.

3.3 Problematik der Evaluierungsmöglichkeiten von ITS

Eine Schwachstelle in der Beurteilung der Effektivität von ITS besteht darin, dass es aufgrund der verschiedenen Umsetzungsmöglichkeiten und dem interaktiven Zusammenspiel der einzelnen Modelle keine einheitliche ITS-Struktur gibt und sich ein Vergleich, auch bezüglich der Qualitätssicherung innerhalb der Umsetzung mehrerer exemplarischer Prototypen, als schwierig erweist (Polson et al., 1988). Es wurden trotz langer Historie von ITS nur exemplarisch empirische, kontrollierte Evaluierungsstudien für einzelne ITS durchgeführt. Shute und Regian (1993) merken an, dass es bei der Veröffentlichung von Tutor-Evaluationen zu einer Auswahlverzerrung gekommen sein kann, obwohl diese ‚gescheiterten' Studien wichtige Informationen für eine Optimierung der Systeme beinhalten würden. Die Autor*innen betonen die Schwierigkeit der Umsetzung von kontrollierten Evaluationsstudien im Schulkontext bezogen auf mangelnde interne Validität und betonen zugleich den Trade-Off, die Erkenntnisse der Labortestungen nicht effektiv auf die Anwendungspraxis generalisieren zu können. Daher werden für diesen konkreten Anwendungsfall sieben Prinzipien zur Evaluierung von ITS präsentiert, die darauf basieren, einen schrittweisen Übergang vom Labor-Setting in die natürliche Umgebung mit steigender externer Validität zu vollziehen. Diese beinhalten u. a. eine genaue Zieldefinition des ITS, geeignete Operationalisierung mithilfe validierter Messinstrumente, Pilotierungen und Determinierung der Datenanalyse. Verschiedene Designs können unterschiedlich angemessen für den Forschungszweck sein und sollen deshalb detailliert abgewägt werden (Shute & Regian, 1993).

Nach Littman und Soloway (1988) existieren externale und internale Evaluierungsansätze. External soll beurteilt werden, ob fehlende oder missverstandene Konzepte tatsächlich gelernt wurden und sich entsprechend in einer besseren Performanz in den Problemlöse-Aufgaben manifestieren. Hierbei kann auch der vom System prognostizierte Lernfortschritt bzw. Wissensstand der Lernenden mit der tatsächlichen Performanz verglichen werden. Internal soll die ITS-Architektur und ihre einzelnen Komponenten analysiert werden, um erklären zu können, weshalb sich das System in welchen konkreten Situationen in der gezeigten Form verhält.

Konkret könnte ein klassisches randomisiertes Experiment durchgeführt werden, bei dem ein Teil der Lernenden mit einem ITS arbeitet und der andere Teil mit einfacheren Methoden auf herkömmlichen E-Learning-Plattformen. Mithilfe von Pre- und Posttests kann der Lernfortschritt miteinander

verglichen werden. Zusätzlich zu behavioral gezeigten Leistungen sollten auch subjektive Einschätzungen der Lernenden erhoben werden, die sich womöglich (noch) nicht im Verhalten zeigen. Insbesondere sind Transferaufgaben im Posttest für die Evaluierung von verfestigtem Wissen von Bedeutung. Da bei diesem sehr simplen Aufbau eines Experiments allerdings viele potenzielle Störvariablen auftreten, benötigt es Leitlinien für das höchste Ausmaß an Parallelität und Kontrolle möglicher Einflussfaktoren.

Obwohl vereinzelt aufgezeigt wurde, dass die Lehreffektivität von ITS nahezu jener von menschlichen Tutor*innen gleichen kann (VanLehn, 2011), ist die Grundintention dennoch, dass alle ITS durch menschliche Lehrkräfte ergänzt werden (Anderson, 1988) bzw. ITS als Ergänzung einer menschlichen Lehrumgebung genutzt werden sollen. Wie diese Systeme in ein bestehendes Lehrsystem eingebaut werden können und welche Voraussetzungen dafür zu berücksichtigen sind, wird im Folgenden aufgezeigt.

4 Rahmenbedingung für eine erfolgreiche Umsetzung in der Hochschullehre

Für die erfolgreiche Umsetzung eines ITS in der Hochschullehre gibt es sowohl Rahmenbedingungen, die erfüllt sein müssen, als auch solche, die erfüllt sein sollten.

Da ITS den Wissensstand der einzelnen Studierenden im Lernenden-Modell speichern müssen, um individuelle Rückmeldung geben zu können, findet, sobald das ITS eine zentrale Datenbank für diese Daten verwendet, die EU-Datenschutz-Grundverordnung (EU-DSGVO) Anwendung. Somit muss zum Beispiel das Einverständnis der Studierenden eingeholt werden, bei Bedarf Auskunft über die gespeicherten Daten gegeben werden und die Möglichkeit vorhanden sein, die Daten auf Wunsch wieder zu löschen.

Mit dem Ziel, allen Studierenden angepasstes, individuelles Feedback zu geben, sollen ITS als Ergänzung vor allem außerhalb der Präsenzlehre genutzt werden. Hier eröffnen sie einen weiteren Raum für rückgekoppeltes Lernen, wie er sonst nur in Seminaren und Tutorien sowie begrenzt in Vorlesungen geboten werden kann. Damit Studierende selbstbestimmt auf das ITS zugreifen können, ist eine Einbettung in die bestehende Infrastruktur der Hochschule sinnvoll.

Eine direkte Einbettung des von uns geplanten ITS in bestehende Infrastrukturen, wie z. B. die verbreiteten Moodle-basierten Plattformen zur Organisation von Lehrveranstaltungen, ist aufgrund technischer Gegebenheiten nur unter hohem Aufwand möglich. Eine bewährte und ressourcensparende Möglichkeit ist deswegen die Implementation als lokale Anwendung oder noch besser als Webapplikation, wie durch den SQL-Tutor von Mitrovic (2003) gezeigt werden konnte. Die Implementierung als Webapplikation hat außerdem den Vorteil, dass das ITS ohne großen Mehraufwand auch Studierenden anderer Universitäten und Hochschulen zur Verfügung gestellt werden kann. Somit kann die Gruppe an Nutzer*innen leichter vergrößert und Folgeforschung zu Effekten der Nutzung des ITS erleichtert werden. Außerdem ist durch eine Webapplikation ein Zugang unabhängig vom genutzten Betriebssystem möglich. Dies muss gegeben sein, um die Verfügbarkeit des ITS

für die Studierenden unabhängig von der genutzten Hard- und Software zu gewährleisten und somit Chancengleichheit zu erreichen.

Zu den Rahmenbedingungen eines ITS und dessen Einsatz gehört auch die Entscheidung, ob *Gamification* oder *Nudging* eingesetzt wird und ob die Nutzung des ITS oder Erfolge im ITS durch Bonuspunkte für Klausuren oder Teilnoten honoriert werden. All dies hat Einfluss auf motivationale Aspekte der Studierenden und beeinflusst damit die Nutzungsrate und Nutzungsdauer des ITS und somit auch die Lerneffekte der Studierenden (Damgaard & Nielsen, 2018; Dichev et al., 2014). Insbesondere klassische Gamification, also z. B. der Einsatz von virtuellen Punkten, Abzeichen oder Bestenlisten, stellt eher eine extrinsische Motivation dar und kann dadurch negative Effekte auf den Lernerfolg haben (Dichev & Dicheva, 2017; Subhash & Cudney, 2018). Allerdings könnten Nudging-Elemente wie die freiwillige Möglichkeit sich selbst bestimmte Fortschrittsziele zu setzen und daran erinnert zu werden manche Studierende auch dabei unterstützen, gleichmäßiger über das Semester verteilt zu lernen (Damgaard & Nielsen, 2018). Letztlich ermöglichen ITS durch das konstruktive Lernen aus Fehlern und das granulare Feedback, die Studierenden intrinsisch zu motivieren. Damit überwinden ITS negative Aspekte der Gamification (Dichev et al., 2014). Gewisse Nudging-Elemente sollten also mit integriert werden, um die Nutzung des ITS zu erhöhen.

Wie W. B. Johnson (1988) schon aufgeführt hat, ist eine Durchführbarkeitsanalyse vor der Entwicklung eines ITS von hoher Relevanz, um später einen erfolgreichen Übergang in den Lernalltag der Studierenden zu ermöglichen. Dabei ist darauf zu achten, dass auf Seiten der Lehrenden und Lernenden ein Bedarf an einem ITS ersichtlich ist und sich die Wissensdomäne, für die das ITS entwickelt werden soll, nach aktuellem technischen Stand überhaupt für eine Umsetzung eignet. Eine Empfehlung wird hier für Domänen ausgesprochen, die von Natur aus komplex und technisch sind. Nach der Entwicklung sollte der Übergang in die Lehre ebenfalls sorgfältig geplant und begleitet werden, sodass der Beginn einer Nutzung des ITS für Studierende mit möglichst wenig Aufwand verbunden ist und sich das Verhältnis von Aufwand und Nutzen rechtfertigen lässt.

5 Fazit und Diskussion

Im Anwendungskontext für die Einführung in Programmieraufgaben könnte die starke Individualisierungsmöglichkeit von ITS von besonderem Vorteil sein. Wir haben aufgezeigt, dass es interindividuelle Unterschiede in metakognitiven Lösungsstrategien und deren Auswirkungen auf die gezeigte Performanz gibt. Diese Zusammenhänge sollten auch im Rahmen von ITS z. B. beim Lösen von Programmierfertigkeiten empirisch untersucht werden, um wissensbasierte Systeme entsprechend der gewonnenen Kenntnisse anzupassen.

Wir konnten zeigen, dass einige gut funktionierende Systeme existieren. Gleichzeitig scheint der ITS Forschungsprozess sehr prototypisch und exemplarisch geprägt zu sein. Dies hängt auch mit der starken Domänenspezifizität als Natur eines ITS zusammen. Dennoch haben wir aufgezeigt, dass die unterschiedlichen Modellierungsmethoden sowohl Vorteile als auch Nachteile mit sich bringen und noch einige Lücken bezogen auf die Standardisierung von Systemen existieren. Hieran soll unser Beitrag im Forschungsprozess anknüpfen.

Unser Ziel ist es, ein möglichst vielfältiges ITS durch die Kombination bestehender Methoden zu entwickeln, das sich nicht nur auf eine Art des Lernenden-Modells oder auf eine Wissensart beschränkt. Besonders die Empirie in der Lehrpraxis kommt in bisherigen Systemen zu kurz.

Wenn es uns gelingt, die Vielfalt von guten Lehrkräften in ein einziges kohärentes System zu integrieren, scheint die Frage nahezuliegen, ob dieses System einen Unterricht vollständig ersetzen könnte. Wir argumentieren hierbei, dass auf die soziale Interaktion im Unterricht sowie deren Auswirkung auf motivationale und emotionale Aspekte des Lernprozesses nicht verzichtet werden kann und soll. Aufgrund ihres orts- und zeitunabhängigen Charakters können ITS Lehrkräfte allerdings durch individualisiertes Feedback, individuell angepasstes Lerntempo sowie die Möglichkeit zur Behandlung individueller Präferenzen optimal unterstützen. Sie können sie insbesondere dort ergänzen, wo im Unterricht aufgrund von Zeit und Größe von Kursen Abstriche in Bezug auf die Individualisierung des Lehrinhalts gemacht werden müssen.

Um eine stärkere und effektivere Vernetzung zwischen der Forschung und Praxis zu gewährleisten, benötigt es für den Entwicklungszyklus von ITS stärkere Richtlinien, Handlungsanweisungen und ausreichend dokumentierte und geleitete Methoden, um weg von der prototypischen Umsetzung einzelner Systeme hin zu einer breiteren Anwendung zu gelangen. Erst mit zunehmender Anwendung als innere, direkte Feedback-Schleife in der Lehre und daraus gewonnenen Einsichten als äußere Feedback-Schleife für die Forschung können die vielversprechenden Vorteile von ITS zum Tragen kommen.

Danksagung

Diese Forschung wurde finanziell unterstützt durch die Stiftung Innovation in der Hochschullehre und das Projekt *Digitale Kulturen der Lehre entwickeln* (*DiKuLe*).

Literatur

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review*, *89*(4), 369–406. https://doi.org/10.1037/0033-295X.89.4.369

Anderson, J. R. (1988). The expert module. In M. C. Polson, J. J. Richardson, & E. Soloway (Hrsg.), *Foundations of intelligent tutoring systems* (S. 21–53). L. Erlbaum Associates.

Anderson, J. R., Conrad, F. G., & Corbett, A. T. (1989). Skill acquisition and the LISP tutor. *Cognitive Science*, *13*(4), 467–505. https://doi.org/10.1016/0364-0213(89)90021-9

Anderson, J. R., & Reiser, B. J. (1985). The LISP tutor. *Byte*, 10, 159–175.

Atkinson, R. K., Derry, S. J., Renkl, A., & Wortham, D. (2000). Learning from examples: Instructional principles from the worked examples research. *Review of Educational Research*, *70*(2), 181–214. https://doi.org/10.3102/00346543070002181

Brown, J. S., Burton, R. R., & Bell, A. G. (1975). SOPHIE: A step toward creating a reactive learning environment. *International Journal of Man-Machine Studies*, *7*(5), 675–696. https://doi.org/10.1016/s0020-7373(75)80026-5

Burns, H. L., & Capps, C. G. (1988). Foundations of intelligent tutoring systems: An introduction. In M. C. Polson, J. J. Richardson, & E. Soloway (Hrsg.), *Foundations of intelligent tutoring systems* (S. 1–19). L. Erlbaum Associates.

Burton, R. R. (1982). Diagnosing bugs in a simple procedural skill. *Intelligent Tutoring Systems*, 157–184. https://ci.nii.ac.jp/naid/10010708678/

Burton, R. R. (1988). The environment module of intelligent tutoring systems. In M. C. Polson, J. J. Richardson & E. Soloway (Hrsg.), *Foundations of intelligent tutoring systems* (S. 109–142). L. Erlbaum Associates.

Carbonell, J. (1970). AI in CAI: An artificial-intelligence approach to computer-assisted instruction. *IEEE Transactions on Man Machine Systems*, *11*(4), 190–202. https://doi.org/10.1109/tmms.1970.299942

Chen, X. (2018). *Evaluating the effects of adaptively presenting worked examples, erroneous examples and problem solving in a constraint-based tutor*. University of Canterbury. https://doi.org/10.26021/3438

Chen, X., Mitrovic, A., & Mathews, M (2020). Learning from worked examples, erroneous examples, and problem solving: Toward adaptive selection of learning activities. *IEEE Transactions on Learning Technologies*, *13*(1), 135–149. https://doi.org/10.1109/tlt.2019.2896080

Clancey, W. J. (1986). From Guidon to Neomycin and Heracles in twenty short lessons. *AI Magazine*, *7*(3), 40. https://doi.org/10.1609/aimag.v7i3.548

Conati, C. (2009). *Intelligent tutoring systems: New challenges and directions*. https://www.ijcai.org/proceedings/09/papers/012.pdf

Damgaard, M. T., & Nielsen, H. S. (2018). Nudging in education. *Economics of Education Review*, *64*, 313–342. https://doi.org/10.1016/j.econedurev.2018.03.008

Darabi, A., Arrington, T. L., & Sayilir, E. (2018). Learning from failure: a meta-analysis of the empirical studies. *Educational Technology Research and Development*, *66*(5), 1101–1118. https://doi.org/10.1007/s11423-018-9579-9

Dichev, C., & Dicheva, D. (2017). Gamifying education: what is known, what is believed and what remains uncertain: a critical review. *International Journal of Educational Technology in Higher Education*, *14*(1), 1–36. https://doi.org/10.1186/s41239-017-0042-5

Dichev, C., Dicheva, D., Angelova, G., & Agre, G. (2014). From gamification to gameful design and gameful experience in learning. *Cybernetics and information technologies*, *14*(4), 80–100.

Diener, C. I., & Dweck, C. S. (1978). An analysis of learned helplessness: Continuous changes in performance, strategy, and achievement cognitions following failure. *Journal of Personality and Social Psychology*, *36*(5), 451–462. https://doi.org/10.1037/0022-3514.36.5.451

Dochy, F., Segers, M., van den Bossche, P., & Gijbels, D. (2003). Effects of problem-based learning: a meta-analysis. *Learning and Instruction*, *13*(5), 533–568. https://doi.org/10.1016/S0959-4752(02)00025-7

Frese, M., & Altmann, A. (1989). The treatment of errors in learning and training. In L. Bainbridge, & S. A. Ruiz Quintanill (Hrsg.), *Developing Skills with Information Technology* (S. 65–86). Wiley.

Gentner, D., & Maravilla, F. (2017). Analogical reasoning. In *The Routledge international handbook of thinking and reasoning* (S. 186–203). Routledge.

Gentner, D., & Markman, A. B. (1997). Structure mapping in analogy and similarity. *American Psychologist*, *52*(1), 45–56. https://doi.org/10.1037/0003-066X.52.1.45

Giacomazzi, E. S., Mäckel F., & Fruth, L. P. (2021). *Subkraki*. Otto-Friedrich-Universität Bamberg, Kognitive Systeme. https://cogsys.uni-bamberg.de/ITS/

Halff, H. M. (1988). Curriculum and instruction in automated tutors. In M. C. Polson, J. J. Richardson, & E. Soloway (Hrsg.), *Foundations of intelligent tutoring systems* (S. 79–110). L. Erlbaum Associates.

Johnson, W. B. (1988). Pragmatic considerations in research, development, and implementation of intelligent tutoring systems. In M. C. Polson, J. J. Richardson, & E. Soloway (Hrsg.), *Foundations of intelligent tutoring systems* (S. 191–207). L. Erlbaum Associates.

Johnson, W. L., & Soloway, E. (1985). PROUST: Knowledge-based program understanding. *IEEE Transactions on Software Engineering*, *SE-11*(3), 267–275. https://doi.org/10.1109/tse.1985.232210

Kapur, M. (2008). Productive failure. *Cognition and Instruction*, *26*(3), 379–424. https://doi.org/10.1080/07370000802212669

Kapur, M. (2016). Examining productive failure, productive success, unproductive failure, and unproductive success in learning. *Educational Psychologist*, *51*(2), 289–299. https://doi.org/10.1080/00461520.2016.1155457

Kilhamn, C., Bråting, K., & Rolandsson, L. (2021). Teachers' arguments for including programming in mathematics education. In G. A. Nortvedt, N. F. Buchholtz, J. Fauskanger, F. Hreinsdóttir, M. Hähkiöniemi, B. E. Jessen, J. Kurvits, Y. Liljekvist, M. Misfeldt, M. Naalsund, H. K. Nilsen, G. Pálsdóttir, P. Portaankorva-Koivisto, J. Radišić, & A. Wernberg (Hrsg.), *Bringing Nordic mathematics education into the future: Proceedings of Norma 20. The ninth Nordic Conference on Mathematics Education* (S. 169–176). Svensk förening för MatematikDidaktisk Forskning - SMDF. https://www.diva-portal.org/smash/record.jsf?pid=diva2:1541426

Koedinger, K. R., Brunskill, E., Baker, R. S., McLaughlin, E. A., & Stamper, J. (2013). New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine*, *34*(3), 27–41. https://doi.org/10.1609/aimag.v34i3.2484

Kornell, N., Hays, M. J., & Bjork, R. A. (2009). Unsuccessful retrieval attempts enhance subsequent learning. *Journal of experimental psychology. Learning, memory, and cognition*, *35*(4), 989–998. https://doi.org/10.1037/a0015729

Lishinski, A., Yadav, A., Good, J., & Enbody, R. (Sept. 2016). Learning to program: Gender differences and interactive effects of students' motivation, goals, and self-efficacy on performance. In J. Sheard (Hrsg.), *Proceedings of the 2016 ACM Conference on International Computing Education Research* (S. 211–220). Association for Computing Machinery. https://doi.org/10.1145/2960310.2960329

Littman, D., & Soloway, E. (1988). Evaluating ITSs: The cognitive science perspective. In M. C. Polson, J. J. Richardson, & E. Soloway (Hrsg.), *Foundations of intelligent tutoring systems* (S. 209–242). L. Erlbaum Associates.

Lorenzet, S. J., Salas, E., & Tannenbaum, S. I. (2005). Benefiting from mistakes: The impact of guided errors on learning, performance, and self-efficacy. *Human Resource Development Quarterly*, *16*(3), 301–322. https://doi.org/10.1002/hrdq.1141

MacMillan, M. (2018). The SoTL literature review: Exploring new territory. In N. L. Chick (Hrsg.), *SoTL in action: Illuminating critical moments of practice* (S. 23–31). Stylus Publishing.

Mathews, M., & Mitrovic, A. (2007). The effect of problem templates on learning in intelligent Tutoring Systems. *13th International Conference on Artificial Intelligence in Education (AIED 2007), 9-13 Jul 2007. Frontiers in Artifical Intelligence and Applications*, *158*, 611–613. http://hdl.handle.net/10092/758

Meier, C. (2002). KI-basierte, adaptive Lernumgebungen. In K. Wilbers, & A. Hohenstein (Hrsg.), *Handbuch E-Learning: Expertenwissen aus Wissenschaft und Praxis* (S. 1–21). Deutscher Wirtschaftsdienst. https://www.alexandria.unisg.ch/257285/

Miller, J. R. (1988). The role of human-computer interaction in Intelligent Tutoring Systems. In M. C. Polson, J. J. Richardson, & E. Soloway (Hrsg.), *Foundations of intelligent tutoring systems* (S. 143–189). L. Erlbaum Associates.

Miller, M. L. (1979). A structured planning and debugging environment for elementary programming. *International Journal of Man-Machine Studies*, *11*(1), 79–95. https://doi.org/10.1016/s0020-7373(79)80006-1

Mitrovic, A. (2003). An intelligent SQL tutor on the web. *International Journal of Artificial Intelligence in Education*, *13*(2-4), 173–197. https://content.iospress.com/articles/international-journal-of-artificial-intelligence-in-education/jai13-2-4-03

Moreno-León, J., Robles, G., Román-González, M., & García, J. D. R. (2019). *Not the same: a text network analysis on computational thinking definitions to study its relationship with computer programming*. *Revista Interuniversitaria de Investigación en Tecnología Educativa (RIITE)*, *7*, 26–35. http://dx.doi.org/10.6018/riite.397151

Najar, A. S., Mitrovic, A., & Neshatian, K. (2014). Utilizing eye tracking to improve learning from examples. In C. Stephanidis, & M. Antona (Hrsg.), *Universal access in human-computer interaction: Universal access to information and knowledge* (S. 410–418). Springer. https://doi.org/10.1007/978-3-319-07440-5_38

Nwana, H. S. (1990). Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, (4), 251–277.

Ohlsson, S. (1992). Constraint-based student modeling. In J. E. Greer, & G. I. McCalla (Hrsg.), *Student modelling: The key to individualized knowledge-based instruction* (S. 167–189). Springer.

Pirolli, P. L., & Anderson, J. R. (1985). The role of learning from examples in the acquisition of recursive programming skills. *Canadian Journal of Psychology/Revue canadienne de psychologie*, *39*(2), 240–272. https://doi.org/10.1037/h0080061

Polson, M. C., Richardson, J. J., & Soloway, E. (1988). *Foundations of intelligent tutoring systems*. L. Erlbaum Associates.

Recker, M. M., & Pirolli, P. (1995). Modeling individual differences in students' learning Strategies. *Journal of the Learning Sciences*, *4*(1), 1–38. https://doi.org/10.1207/s15327809jls0401_1

Renkl, Stark, Gruber, & Mandl (1998). Learning from worked-out examples: The effects of example variability and elicited self-explanations. *Contemporary educational psychology*, *23*(1), 90–108. https://doi.org/10.1006/ceps.1997.0959

Rodrigues, M., Novais, P., & Santos, M. F. (2005). *Future challenges in intelligent tutoring systems: A framework*. Formatex Research Center. https://repositorium.sdum.uminho.pt/handle/1822/3174

Shah, P., Berges, M., & Hubwieser, P. (2017). Qualitative content analysis of programming errors. *Proceedings of the 5th International Conference on Information and Education Technology*, 161–166. https://doi.org/10.1145/3029387.3029399

Shute, V. J., & Regian, J. W. (1993). Principles for evaluating Intelligent Tutoring Systems. *Journal of Artificial Intelligence in Education, 4*(2-3), 245–271.

Steenhof, N., Woods, N. N., & Mylopoulos, M. (2020). Exploring why we learn from productive failure: insights from the cognitive and learning sciences. *Advances in Health Sciences Education*, *25*(5), 1099–1106. https://doi.org/10.1007/s10459-020-10013-y

Steuer, G. (2014). Umgang mit Fehlern auf individueller Ebene. In G. Steuer (Hrsg.), *Fehlerklima in der Klasse: Zum Umgang mit Fehlern im Mathematikunterricht* (S. 33–48). Springer VS. https://doi.org/10.1007/978-3-658-05293-5_4

Subhash, S., & Cudney, E. A. (2018). Gamified learning in higher education: A systematic review of the literature. *Computers in Human Behavior*, *87*, 192–206. https://doi.org/10.1016/j.chb.2018.05.028

Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. *Learning and Instruction*, *4*(4), 295–312. https://doi.org/10.1016/0959-4752(94)90003-5

Sweller, J., & Cooper, G. A. (1985). The use of worked examples as a substitute for problem Solving in learning algebra. *Cognition and Instruction*, *2*(1), 59–89. https://doi.org/10.1207/s1532690xci0201_3

Tulis, M., Steuer, G., & Dresel, M. (2016). Learning from errors: A model of individual processes. *Frontline Learning Research*, *4*(2), 12–26. https://eric.ed.gov/?id=ej1108798

VanLehn, K. (1988). Student modeling. In M. C. Polson, J. J. Richardson, & E. Soloway (Hrsg.), *Foundations of intelligent tutoring systems* (S. 55–78). L. Erlbaum Associates.

VanLehn, K. (1998). Analogy events: How examples are used during problem solving. *Cognitive Science*, *22*(3), 347–388. https://doi.org/10.1207/s15516709cog2203_4

VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, *16*(3), 227–265. https://content.iospress.com/articles/international-journal-of-artificial-intelligence-in-education/jai16-3-02

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, *46*(4), 197–221. https://doi.org/10.1080/00461520.2011.611369

Yew, E. H., & Goh, K. (2016). Problem-based learning: An overview of its process and impact on learning. *Health Professions Education*, *2*(2), 75–79. https://doi.org/10.1016/j.hpe.2016.01.004

Zeller, C., & Schmid, U. (2017). *Automatic generation of analogous problems to help resolving misconceptions in an intelligent tutor system for written subtraction*. [Vortrag]. 24th International Conference on Case Based Reasoning, Atlanta, GA, 31th October-2nd November 2016. opus. https://fis.uni-bamberg.de/handle/uniba/41882

Zinn, C. (2013). Algorithmic debugging for intelligent tutoring: How to use multiple models Improve diagnosis. In I. J. Timm, & M. Thimm (Hrsg.), *KI 2013: Advances in Artificial Intelligence* (S. 272–283). Springer. https://doi.org/10.1007/978-3-642-40942-4_24