12 Erste Schritte zur automatisierten Generation von Items in einem webbasierten Tracingsystem

Morten Bastian und Andreas Mühling

Inhaltsverzeichnis

M. Bastian (✉) · A. Mühling
Institut für Informatik, Universität Kiel, Kiel, Deutschland
E-Mail: mba@informatik.uni-kiel.de

A. Mühling
E-Mail: andreas.muehling@informatik.uni-kiel.de

J. Roth et al. (Hrsg.), *Die Zukunft des MINT-Lernens – Band 1*,
https://doi.org/10.1007/978-3-662-66131-4_12

12.1 Einleitung

Programmieren ist eine Kompetenz, die in den letzten Jahren immer mehr an Bedeutung gewinnt. Ein Beispiel dafür ist, dass es explizit als eine Fähigkeit für das lebenslange Lernen aufgezählt wird (Europäische Union, 2018). Eng verknüpft mit dem Programmieren und auch im schulischen Kontext international relevant ist das Computational Thinking.

Computational Thinking
Computational Thinking bezieht sich auf die Fähigkeit einer Person, Aspekte realweltlicher Probleme zu identifizieren, die für eine (informatische) Modellierung geeignet sind, algorithmische Lösungen für diese (Teil-)Probleme zu bewerten und selbst so zu entwickeln, dass diese Lösungen mit einem Computer operationalisiert werden können. Die Modellierungs- und Problemlösungsprozesse sind dabei von einer Programmiersprache unabhängig (vgl. Fraillon et al., 2020).

Die damit verbundenen Fähigkeiten werden zukünftig auch vermehrt in Studium und Beruf relevant sein (Weintrop et al., 2016) und sind daher inzwischen auch Teil der *International Computer and Literacy Study* (Fraillon et al., 2020).

Häufig werden die Konzepte des Computational Thinking durch das Programmieren von einfachen Systemen, die auf Grundkonzepte des Programmierens abzielen, gelehrt (Rose, 2019). In der Programmierung – wenigstens in imperativen Sprachen – werden einzelne Anweisungen durch die sogenannten Kontrollstrukturen Sequenz, Wiederholung und Bedingung zu komplexen Programmen zusammengefügt (Rose, 2019).

Diese Bausteine von Programmen sind syntaktisch und semantisch eindeutig definiert, nur deswegen ist ein Programm von einer Maschine ausführbar. Daher eignen sich Programmieraufgaben sehr gut für eine automatisierte Bewertung (Keuning et al., 2019) und mutmaßlich auch für eine automatisierte Diagnostik von zum Beispiel typischen Fehlvorstellungen. Seit Langem sind verschiedene Fehlvorstellungen bekannt, die bei dem Umsetzen einer Problemlösung als Programm auftreten (z. B. Pea, 1986; Sorva, 2012).

Progly[1] ist ein webbasiertes Testsystem, das die Durchführung von sogenannten Tracingaufgaben (siehe unten) erlaubt (Bastian et al., 2021). Durch die im System gesammelten Daten können Rückschlüsse auf Fehlvorstellungen gezogen werden,

[1] Bei Interesse an dem Testsystem wenden Sie sich bitte direkt an die Autoren. Es ist geplant, das System in der Zukunft öffentlich bereitzustellen.

die Auswertung der Aufgaben erfolgt automatisch. In diesem Beitrag präsentieren wir die Ergebnisse einer Studie, die das Ziel verfolgt, neue Erkenntnisse im Kontext der Itemkonstruktion für Tracingaufgaben zu gewinnen. Dabei liegt der Fokus auf dem Erzeugen von Items mit einer vorhersagbaren Schwierigkeit sowie einer Ausweitung des Itempools durch Modifikationen von Items, ohne dabei deren wesentliche schwierigkeitsgenerierende Merkmale zu verändern.

12.2 Bisherige Forschung

12.2.1 Tracing als Vorläuferfähigkeit

Empirisch belegt ist eine Serie von Entwicklungsschritten, die Personen beim Programmierenlernen durchlaufen (Lopez et al., 2008). Während sie zunächst nur in der Lage sind, einzelne Programmzeilen in ihrer Auswirkung zu erfassen, erweitert sich das Verständnis über zunächst Blöcke bis hin zu ganzen Programmen. Dem zugeordnet sind die Fertigkeiten des Tracings, des Erklärens und schließlich des eigenständigen Schreibens von Programmen bzw. Programmfragmenten. Das Tracing ist somit eine erste Fertigkeit, die Lernende entwickeln und die man überprüfen kann (Lopez et al., 2008).

Tracing
Tracing beschreibt eine schrittweise, gedankliche Ausführung eines konkreten Programmablaufs und die Fähigkeit, die Auswirkung eines Programmschritts, insbesondere den nächsten auszuführenden Schritt, bestimmen zu können (vgl. Perkins et al., 1986).

Die Relevanz des Tracings nimmt auch mit fortschreitender Expertise nicht ab. Im Besonderen die Tätigkeit des Debuggings wird von dieser Fertigkeit beeinflusst. Debugging bezeichnet die erfolgreiche, typischerweise systematische Identifikation und Behebung von Fehlern in einem Programm. Das Fehlen einer ausgeprägten Tracingfähigkeit führt zu geringeren Leistungen im Debugging (Lister et al., 2004).

Beim Tracing eines Programms durch eine Person kann es zu Abweichungen von der korrekten Folge der während eines Programmablaufs ausgeführten Anweisungen – im Rahmen des Beitrags verwenden wir hierfür im Folgenden den englischen Begriff *Trace* – kommen. Neben Flüchtigkeitsfehlern können aufgrund der semantisch eindeutigen Definition von Programmen und ihren Bausteinen diese Abweichungen nur durch fehlendes oder falsches Wissen (im Sinne einer Fehlvorstellung) über die Funktionsweise ausgelöst werden.

Ein Beispiel dafür ist das *vorzeitige Abbrechen einer Wiederholung mit Abbruchbedingung* (Pea, 1986). Es konnte gezeigt werden, dass diese Fehlvorstellungen bei Lernenden unabhängig von der genutzten Programmiersprache und des Alters auftreten können (Pea, 1986; Sorva, 2012).

Tracingaufgaben sind somit – speziell, aber nicht ausschließlich – im Anfangsunterricht zum Thema Programmieren eine sehr wertvolle Informationsquelle für Lehrende, da aus falschen Antworten sehr schnell und sehr spezifisch auf bekannte Lernprobleme geschlossen werden kann. Daher bilden sie die Grundlage des in diesem Beitrag verwendeten Testsystems. Gleichzeitig ist die Struktur typischer Tracingaufgaben sehr einfach und sie eignen sich somit gut für eine automatische Generierung. Dies begünstigt eine einfache Skalierbarkeit eines existierenden Testsystems, da der Itempool damit auch bei wiederholter Anwendung, zum Beispiel als Lernverlaufsdiagnostik, nicht erschöpft wird (Klauer, 2014). Gestaltet man den Test darüber hinaus adaptiv, können diagnostische Informationen sehr zeitökonomisch im Unterricht gesammelt werden (Frey, 2012).

12.2.2 Wege zur Bestimmung der Itemschwierigkeit eines Messinstruments

Für eine perspektivisch automatische Generierung von Items mit spezifischen Eigenschaften ist es nötig, ein Maß für die Itemschwierigkeit von Tracingaufgaben zu haben, das sich auch auf potenziell noch unbekannte Items anwenden lässt. Zur Bestimmung der Itemschwierigkeit existieren grundsätzlich verschiedene Möglichkeiten (vgl. Choi & Moon, 2020; Moosbrugger & Kelava, 2012; Duran et al., 2018):

1. Normativ durch die Bewertung von Expertinnen und Experten,
2. Empirisch durch die Pilotierung von Items,
3. Durch analytische Indikatoren, die für die (theoretische) Bestimmung der Schwierigkeit ausgewertet werden können.

Speziell für den hier relevanten Kontext hat die dritte Vorgehensweise den Vorteil, dass sie in einem System implementiert und damit ad hoc für neue Items angewendet werden kann. Die empirische Ermittlung kann hingegen nur post hoc angewendet werden. Eine normative Bewertung kann grundsätzlich ebenfalls nur post hoc stattfinden. Wenn feste Bewertungskriterien der Expertinnen und Experten bekannt sind, wäre es jedoch denkbar, diese zu implementieren und im Sinne von Indikatoren ad hoc für eine Einschätzung der Schwierigkeiten zu berücksichtigen.

Im Rahmen einer Vorstudie (siehe Abschn. 12.4) werden die drei Verfahren anhand von gegebenen Items evaluiert. Weitergehend wird sich der Forschungsfrage gewidmet, ob auf Basis der ermittelten Rangfolgen der Schwierigkeiten ein Konsens – im Sinne einer Regelmenge –, der für die Konstruktion von neuen Items einer bestimmten Schwierigkeit nutzbar ist, ermittelt werden kann.

12.3 Testsystem

Das genutzte Testsystem ist eine digitale Weiterentwicklung eines psychometrischen Tests (Mühling et al., 2015). In einer Vorstudie konnte die Validität des hier als Grundlage genutzten Messinstruments bestätigt werden und es konnten mit dem Testsystem 24 Fehlvorstellungen in 5 übergeordneten Kategorien identifiziert werden (Bastian et al., 2021). Der in der vorangegangenen Studie genutzte Test besteht aus insgesamt 9 Items, die in einer festen Reihenfolge präsentiert werden. In diesem Beitrag werden diese (alten) Items als A1–A9 bezeichnet.

Zur Durchführung einer Messung wird keine spezielle Software, sondern lediglich ein Browser und eine Internetverbindung benötigt. Das Design des Testsystems erlaubt eine Durchführung am PC, Laptop oder Tablet.

Während einer Messung wird für jedes Item die Umgebung aus Abb. 12.1 angezeigt. Sie besteht aus einem 8×8 Feld, einem vorgegebenen Programm, den

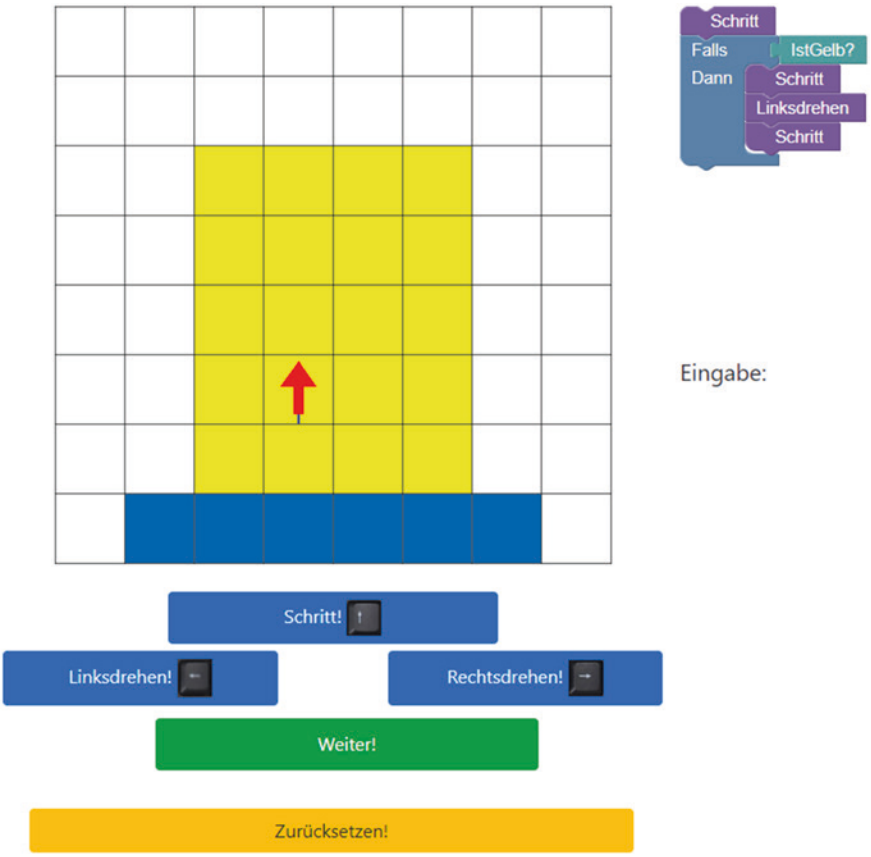

Abb. 12.1 Startdarstellung des Items I4 (siehe Abschn. 12.4.1)

Kontrollknöpfen zum Bewegen der Figur, einem „Zurücksetzen!“-Knopf und einer Anzeige bzgl. der eingegebenen Sequenz. Bewegungen können mittels der Knöpfe oder der Tastatur erfolgen. Die bewegliche Figur wird durch einen roten Pfeil gekennzeichnet. Die Pfeilspitze gibt an, in welche Richtung die Figur „blickt“. Vor der Messung werden alle Blöcke, die während einer Messung vorkommen, erklärt. Die Blöcke orientieren sich dabei an üblichen Programmiersprachen, sind aber von der expliziten Syntax einer speziellen Programmiersprache abstrahiert. Zur Darstellung der Blöcke wird die freie Bibliothek „Blockly“[2] verwendet, sodass die Darstellung eine Ähnlichkeit zu typischen visuellen Programmiersprachen, wie zum Beispiel Scratch (Resnick et al., 2009), aufweist.

Die gestellte Aufgabe besteht darin, die Figur genauso zu bewegen, wie es das vorgegebene Programm angibt. Dabei können die teilnehmenden Personen die Eingabe jederzeit zurücksetzen und neu beginnen.

Im Anschluss an den Test folgt eine kurze Befragung zu Alter, Geschlecht und Lernort (Schule/Universität). Am Ende wird die Anzahl der korrekt gelösten Items angezeigt. Während der Messung gab es kein Feedback seitens des Systems. Zur Auswertung der Antworten wird der zuletzt eingegebene Trace auf seine Korrektheit überprüft. Es wird jedoch die gesamte Interaktion – inklusive des Zurücksetzens – für mögliche Analysen gespeichert.

12.4 Vorstudie

Im Rahmen der Vorstudie werden die Schwierigkeiten bereits existierender Items, die im Testsystem umgesetzt wurden, mittels drei unterschiedlicher Verfahren ermittelt. Die Forschungsfrage zur Vorstudie lautet:

Lassen sich anhand empirischer Daten sowie normativer und Indikator-gestützter Bewertung existierender Items schwierigkeitsgenerierende Merkmale ermitteln?

12.4.1 Datenerhebung und Auswertungsmethode

Zur empirischen Ermittlung der Schwierigkeit wurden Daten erhoben und ein Rasch-Modell gefittet (Bastian et al., 2021). Die Ergebnisse dieser Studien hinsichtlich der Rangfolge der Schwierigkeiten der Items A1–A9 werden hier übernommen und durch ein Expertenrating (Hughes, 1996) ergänzt. Dieses wurde von zwei Informatik-Lehrkräften aus Schleswig-Holstein durchgeführt. Die gestellte Aufgabe für die Lehrkräfte bestand darin, die Items gemeinsam nach den von ihnen angenommenen Schwierigkeiten zu sortieren.

[2] Siehe auch: https://developers.google.com/blockly.

Als Indikator-gestütztes Maß wurde das von Duran et al. (2018) vorgestellte *Cognitiv Complexity of Computer Programs (CCCP)* Framework angewandt. Es definiert für ein gegebenes Programm ein hierarchisches Baummodell anhand von festen Regeln. Die nötigen Ebenen um das Programm als Baum darzustellen, ergeben die „Komplexität" des Programms. Wir verwenden dieses Maß als Indikator für die Itemschwierigkeit, relevant ist hierbei nicht der absolute Wert, sondern der relative Vergleich der Werte verschiedener Items. Ein Beispiel für solch einen Baum ist für das vorgestellte Beispiel-Item I4 (Abb. 12.1) in Abb. 12.2 dargestellt.

Um einen „Konsens" zwischen diesen drei Verfahren zu bilden, wurden die Ergebnisse aggregiert, indem jedem Item anhand der Position in der jeweiligen Rangfolge eine Kennzahl von 1 bis 9 bzw. im CCCP-Framework aufgrund gleicher Komplexitäten von 1 bis 4 zugeordnet wird. Daraufhin wurden diese Werte gewichtet, indem sie durch die Summe der vergebenen Werte geteilt wurden (bei Rasch und dem Expertenrating durch 45 und bei CCCP durch 28). Als abschließender Schritt wurden diese Werte summiert.

12.4.2 Ergebnisse

Die empirische Ermittlung der Schwierigkeiten wurde mithilfe des Rasch-Modells durchgeführt (Bastian et al., 2021). Die Rangfolge, die sich dabei ergibt, ist:

A1 < A2 < A6 < A3 < A7 < A8 < A9 < A4 < A5.

Die gemeinsam von den Experten erzeugte Rangfolge ist:

A1 < A2 < A6 < A5 < A3 < A7 < A4 < A9 < A8.

Für das CCCP-Framework gibt es für die Items Fälle, in denen mehrere Items dieselbe Komplexität aufweisen, die erzeugte Rangfolge mit diesem Verfahren ist:

A1 < A2 < A3 = A4 = A8 < A5 = A6 = A7 = A9.

Aus diesen drei Rangfolgen und der gewählten Methode der Aggregation ergibt sich die „gemeinsame" Rangfolge als:

A1 < A2 < A6 < A3 < A7 < A5 < A8 < A4 < A9.

12.4.3 Diskussion

Die Ergebnisse unterscheiden sich zwischen den Verfahren an mehreren Stellen. Gleichzeitig sind alle Verfahren durch spezifische Merkmale beeinflusst: Die empirischen Schwierigkeiten können zum Beispiel durch Flüchtigkeitsfehler beeinflusst sein. Das analytische Maß des CCCP-Modells bezieht diese nicht mit ein, zeigt aber für die einfachen Programmfragmente nur geringe Varianz und kann sich durch gezielte geringe Modifikationen eines Programms verändern, ohne dass diese Veränderungen auch eine gesteigerte empirische Schwierigkeit nahelegen. Die normative Rangfolge basiert wiederum auf den subjektiven Erfahrungen der Lehrkräfte.

Durch die Aggregation der Rangfolgen lässt sich somit möglicherweise ein reliableres Bild der Schwierigkeiten bestimmter Aufgaben ermitteln. Aus der

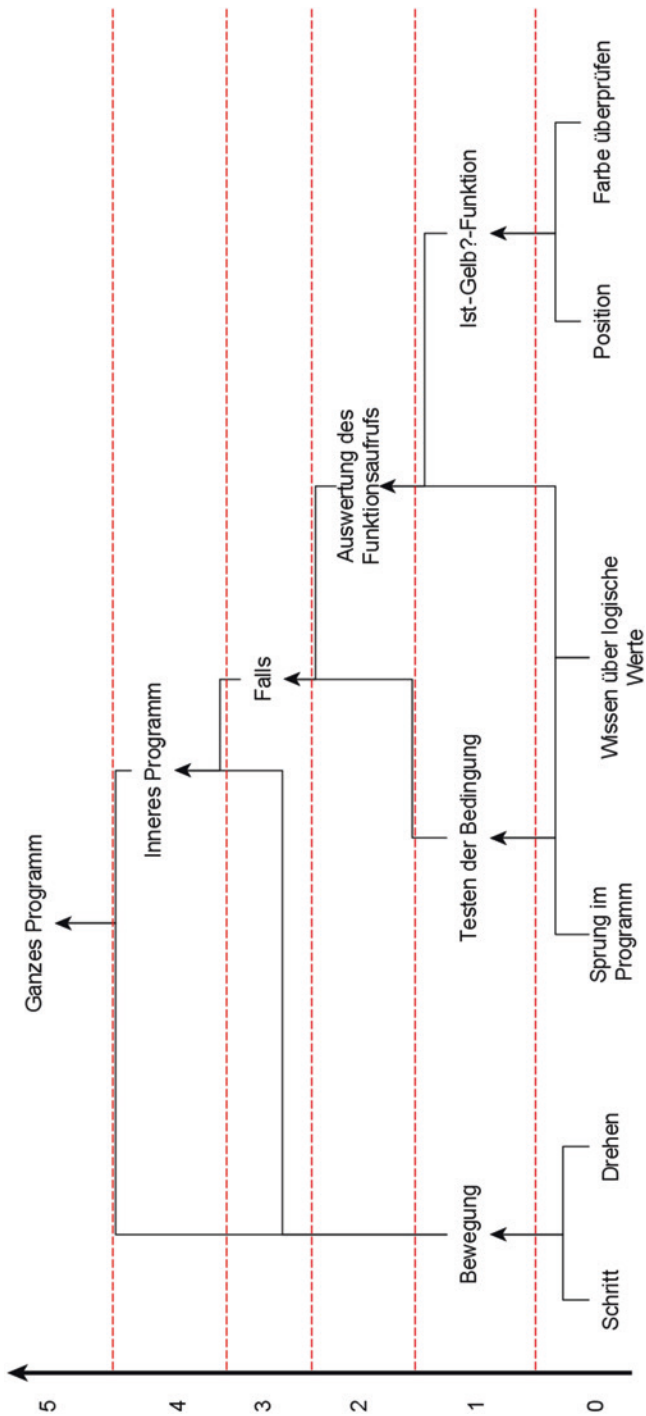

Abb. 12.2 Hierarchischer Baum des Items I4 (Abb. 12.1)

Analyse der aggregierten Rangfolge der Items lassen sich mögliche Regeln für die Generierung von Items formulieren:

Das Konzept der *Wiederholung mit fester Anzahl an Iterationen* (A2) ist leichter als eine *bedingte Anweisung ohne Alternative* (A5), bei der die Bedingung wahr ist und bleibt, die wiederum leichter ist als eine *Wiederholung mit Abbruchbedingung* (A4).

Diese Regeln sollen in der nachfolgenden Studie als Ausgangspunkt für eine empirische Überprüfung der Itemschwierigkeiten verwendet werden.

12.5 Hauptstudie

In der aktuellen Studie soll anhand der vorbereitenden Ergebnisse aus der Vorstudie eine Forschungsfrage explorativ überprüft werden:

Kann man theoriegeleitet anhand von schwierigkeitsgenerierenden Merkmalen Tracingitems erzeugen, die empirisch eine vorhersehbare Schwierigkeit aufweisen?

Es wurden dafür neue Items (I1–I9) konstruiert (Tab. 12.1), die sich im Gegensatz zu den Items der Vorstudie zunächst auf die Konzepte *Wiederholung mit Abbruchbedingung* und *bedingte Anweisung mit und ohne Alternative* fokussieren. Ziel bei der Konstruktion der Items war es, zwei Teilfragen beantworten zu können:

1. Ordnen sich die Schwierigkeiten von neu und vergleichbar zu A2, A5 und A4 konstruierten Items identisch zu den abstrahierten Regeln der Vorstudie an? Vergleichbar heißt in diesem Fall, dass sie zum einen die gleichen Kontrollstrukturen und zum anderen die gleichen Fehlvorstellungen überprüfen.
2. Weisen Items mit leichten Modifikationen, die das analytische Maß verändern, auch empirisch andere Itemschwierigkeiten auf? Leichte Modifikationen

Tab. 12.1 Im Test genutzte Items

Bezeichner	Beschreibung
I1	Sequenz von einfachen Anweisungen
I2	Wiederholung mit einer festen Anzahl an Iterationen
I3	Bedingte Anweisung mit wahrer Bedingung ohne Alternative
I4	Bedingte Anweisung mit wahrer Bedingung ohne Alternative und einer einfachen Anweisung vor der bedingten Anweisung
I5	Bedingte Anweisung mit wahrer Bedingung und mit Alternative
I6	Bedingte Anweisung mit falscher Bedingung und mit Alternative
I7	Wiederholung mit einer Abbruchbedingung
I8	Wiederholung mit einer Abbruchbedingung
I9	Wiederholung mit einer Abbruchbedingung und zwei einfachen Anweisungen nach der Wiederholung

Tab. 12.2 Erwartete Fehlvorstellungen bei den in der Studie genutzten Items

Bezeichner	Fehlvorstellung
I3 & I4	Bedingte Anweisung als Wiederholung
I5	Ausführen der Alternative nach der korrekten Ausführung der Anweisungen im Dann-Fall
I6	Die Alternative der bedingten Anweisung wird wiederholt ausgeführt (Bedingung bleibt unwahr)
I7 & I9	Vorzeitiges Abbrechen der Wiederholung mit Abbruchbedingung
I7 & I8 & I9	Wiederholung mit Abbruchbedingung wird als bedingte Anweisung ohne Alternative ausgeführt

beinhalten in diesem Kontext 1) das Hinzufügen von einfachen Anweisungen vor oder nach einer bedingten Anweisung oder Wiederholung mit Abbruchbedingung, 2) das Verändern der einfachen Anweisungen innerhalb einer Wiederholung bzw. bedingten Anweisung und 3) das Verändern der Anzahl an einfachen Anweisungen innerhalb einer Wiederholung bzw. bedingten Anweisung oder die Umpositionierung der Figur.

Item I2, I4, I7 und I9 wurden so konstruiert, dass sie vergleichbar mit den Items A2, A4 und A5 der Vorstudie (Bastian et al., 2021) sind. Erwartet wird, dass I2 auf jeden Fall vor I4 und diese beiden vor I7 bzw. I9 in der Rangfolge auftauchen. Zudem wurden die Itempaare I3 und I4, I5 und I6 sowie I7 und I8 bzw. I9 jeweils so konstruiert, dass sie im Vergleich zueinander leichte Modifikationen aufweisen.

Die Fehlvorstellungen, die durch die jeweiligen Items überprüft werden sollen, sind in Tab. 12.2 angegeben.

12.6 Aktuelle Studie

12.6.1 Stichprobe

Die Überprüfung der Forschungsfrage erfolgt im Rahmen einer Studie, die an drei Schulen in Norddeutschland und mit Studierenden an der Christian-Albrechts-Universität zu Kiel im Rahmen einer Informatik-Erstsemesterveranstaltung durchgeführt wurde.

Insgesamt haben 273 Personen an der Studie teilgenommen. Zwei Personen wurden aus den Analysen entfernt, da sie keine Eingaben getätigt haben. Die Durchführung des Tests (inklusive des Lesens der Einführung) hat – ohne Berücksichtigung von 11 Personen mit Zeitwerten von mehreren Stunden – im Mittel 6,49 min gedauert.

Aus den Schulen haben 202 Personen (w: 60, m: 131 und 11 divers oder keine Angabe) an der Studie teilgenommen. Das mittlere Alter der teilnehmenden

Personen beträgt zum Zeitpunkt der Studie 13,2 (SD: 1,9; $N = 195$). 7 Personen haben unrealistische Angaben (Alter ≥ 20) getätigt. Da von ihnen aber sinnvolle Traces während der Messung erzeugt wurden, wurden diese Personen nicht für die Analysen entfernt.

Aus der Universität haben 69 Personen (w: 19, m: 44 und 6 divers oder keine Angabe) an der Studie teilgenommen. Das mittlere Alter der teilnehmenden Personen beträgt zum Zeitpunkt der Studie 21,4 (SD: 4,0).

Die Studie wurde im Rahmen des regulären Unterrichts in der Schule und in Eigenarbeit an der Universität im Zeitraum vom 06.12.2021 bis zum 14.01.2022 durchgeführt.

12.6.2 Auswertungsmethode

Zur Überprüfung der Forschungsfrage in der aktuellen Studie wird auf die latente Modellierung mittels des in bildungswissenschaftlichen Studien üblichen Rasch-Modells zurückgegriffen (Bartholomew et al., 2008). Untersucht werden neben der EAP- und WLE-Reliabilität auch der In- und Outfit der Items. Eine weiterführende Analyse der Differenzen der relevanten Itemschwierigkeiten (siehe Abschn. 12.5) wird mittels Chi-Quadrat-Tests überprüft. Alle Analysen erfolgten mit GNU-R. Das Rasch-Modell wurde unter der Verwendung des Pakets TAM (Version: 3.6–45) gefittet.

Die Traces werden auf die erwarteten Fehlvorstellungen untersucht. Die Einordnung erfolgt dabei automatisiert und anhand der Ergebnisse der vorherigen Studie (Bastian et al., 2021).

12.6.3 Ergebnisse

Für die Auswertungen der Daten wurde das Item I1 aus den Datensätzen entfernt, da es als Probe-Item für einen Einstieg in den Test und das zu nutzende System gedacht war. Es sollte somit eine „Eisbrecherfunktion“ erfüllen (Moosbrugger & Kelava, 2012).

Abb. 12.3 stellt die Wright Map des Modells dar. Das Modell weist eine gute EAP- und WLE-Reliabilität mit EAP: 0,73 und WLE: 0,61 auf. Einzig die In- und Outfit-Werte für das Item I4 weisen erhöhte Werte auf. In der Tab. 12.3 sind die jeweiligen Schwierigkeiten und der In- und Outfit für die jeweiligen Items angeben.

Eine Überprüfung der Differenzen bzgl. der relevanten Schwierigkeitspaarungen kann in der Tab. 12.4 eingesehen werden. Lediglich für die Paarung I3 und I4 zeigt sich keine Signifikanz.

Insgesamt wurden alle 974 falschen der insgesamt 2168 Traces auf ihre Fehlerursachen überprüft. Zuvor noch nicht beschriebene Fehler – aufgrund der geänderten Items – wurden im Autorenteam diskutiert und manuell codiert. Dabei handelt es sich um 221 inkorrekte Traces, die in drei Fehlerarten eingeordnet

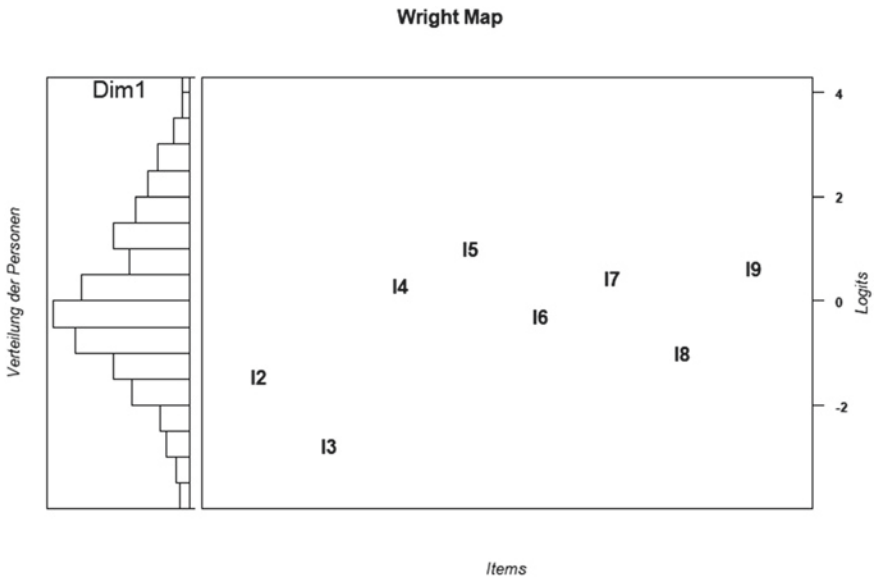

Abb. 12.3 Wright Map des ermittelten Rasch-Modells

Tab. 12.3 Ermittelte Schwierigkeit, Infits und Outfits der Items

Bezeichner	Schwierigkeit	Infit	Outfit
I2	-1,44	1,09	1,27
I3	-2,76	1,02	1,28
I4	0,31	1,43	1,69
I5	1,02	0,87	0,77
I6	-0,28	0,94	0,85
I7	0,46	0,85	0,81
I8	-1,00	0,93	0,87
I9	0,63	0,84	0,76

Tab. 12.4 Ermittelte p-Werte der Chi-Quadrat-Tests für die relevanten Vergleiche der Schwierigkeiten

Item 1	Item 2	χ^2 (df = 1)	p
I3	I4	0,73	0,39
I5	I6	75,92	< 0,001
I7	I8	79,19	< 0,001
I7	I9	139,58	< 0,001
I8	I9	68,98	< 0,001

Tab. 12.5 Beispiele bisher nicht beschriebener Fehlvorstellungen bei den Items I5 und I6

Fehlvorstellung	Korrekter Trace	Falscher Trace
Alternative ausgeführt, ggf. auch als Wiederholung	M, R, M	M, R, M, L (beliebig viele L können folgen)
Als Wiederholung ausgeführt	R, M, L	R, M, L, R, M, L, M, M
Konstrukt ignoriert und sequentiell ausgeführt	R, M, L	L, M, M, R, M, L

wurden. Beispiele dafür sind in der Tab. 12.5 und die zugehörigen Items in der Abb. 12.4a und Abb. 12.4b einzusehen.

Von den falschen Traces wurden 636 als bekannte Fehlvorstellungen kategorisiert. Die restlichen 338 Traces ordnen sich in die Kategorien Flüchtigkeitsfehler (186), Muster nicht erkennbar (93 Vorkommen) oder keine Eingabe getätigt (59 Vorkommen) ein. Beispiele für Flüchtigkeitsfehler sind das Übersehen einer Anweisung vor einem Konzept, das zusätzliche Ausführen einer Anweisung nach der korrekten Ausführung des Programms oder das einmal zu seltene oder zu häufige Ausführen einer Wiederholung mit fester Anzahl an Iterationen.

Die entdeckten Fehlvorstellungen und die zugehörigen Items können in der Tab. 12.6 eingesehen werden.

12.7 Diskussion

Die Rasch-Skalierung der neu generierten Items zeigt eine gute Reliabilität und auch die Itemkennwerte weisen darauf hin, dass der Test mit den neu konstruierten Items weiterhin ein eindimensionales latentes Konstrukt misst. Für eine potenzielle, automatische Itemkonstruktion ist das ein wertvolles Indiz, um Items anhand eines Regelsatzes zu erzeugen. Einzig Item I4 zeigt Auffälligkeiten in seinen In- und Outfit-Werten. Häufige Fehler bei diesem Item umfassen die Fehlvorstellung *bedingte Anweisung ohne Alternative wird als Wiederholung durchgeführt* (69 Vorkommen) und zwei Flüchtigkeitsfehler (gesamt 62 Vorkommen). Die Flüchtigkeitsfehler sind *Übersehen einzelner Schritt-Anweisung vor der bedingten Anweisung* und *einzelne Schritt-Anweisung zu viel am Ende des Traces*. Durch diese drei Fehler lassen sich 131 der 151 inkorrekten Traces beschreiben. Eine mögliche Erklärung für die auffälligen Werte könnte also in der hohen Anzahl an Flüchtigkeitsfehlern liegen.

Im Rahmen der Studie sollte eine Forschungsfrage bestehend aus zwei Teilfragen (siehe Abschn. 12.5) überprüft werden. Dafür wurde eine Rangfolge anhand von drei unterschiedlichen Verfahren zur Einschätzung der Itemschwierigkeit aggregiert und basierend darauf neue Items konstruiert. Die Daten bestätigen wie angenommen die Rangfolge der Schwierigkeiten hinsichtlich der zwei Arten von Wiederholungen und der bedingten Anweisung.

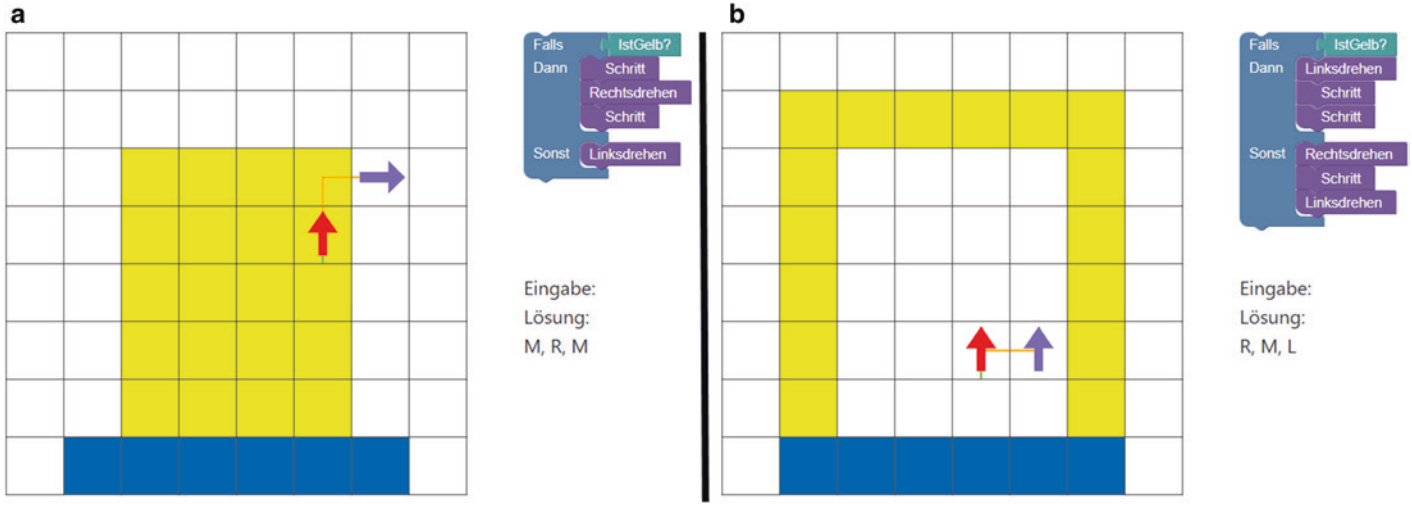

Abb. 12.4 **a** Darstellung des Items I5 mit dem korrekt ausgeführten Trace. **b** Darstellung des Items I6 mit dem korrekt ausgeführten Trace

Tab. 12.6 Übersicht der gefundenen Fehlvorstellung bei den Items

Beschreibung	Vorkommen in falschen Traces bei Item X
Bedingte Anweisung ohne Alternative wird als Wiederholung durchgeführt	I3: 17 von 31 I4: 69 von 151
Alternative wird fälschlicherweise ausgeführt	I5: 135 von 183
Alternative wird als Wiederholung ausgeführt	I5: 17 von 183
Bedingte Anweisung mit Alternative als Wiederholung	I6: 60 von 123
Bedingte Anweisung ignoriert und beide Fälle sequentiell nacheinander	I6: 38 von 123
Vorzeitiges Abbrechen einer Wiederholung mit Abbruchbedingung	I7: 63 von 158 I9: 36 von 158
Abbrechen einer Wiederholung aufgrund von Intentionalität	I7: 11 von 158 I9: 17 von 166
Wiederholung als bedingte Anweisung	I7: 46 von 158 I8: 53 von 90 I9: 74 von 166

Für die zweite Teilfrage, ob leichte Modifikationen von Items in der empirischen Schwierigkeit nachgewiesen werden können, wurden explizit die Paarungen der Items I3 und I4, I5 und I6 sowie I7, I8 und I9 konstruiert.

Für fast alle Paarungen sind die Chi-Quadrat-Tests signifikant. Für Item I5 und I6 bedeutet dies, dass das Hinzufügen von mehr Anweisungen als bei Item I6, die Umpositionierung der Figur und das Ändern der Anweisungen keinen Einfluss auf die Schwierigkeit hatte. Die Umpositionierung der Figur, sodass in Item I5 der Dann-Fall ausgeführt wird und in Item I6 der Sonst-Fall, hatte auch keinen signifikanten Einfluss auf die Schwierigkeit.

I3 und I4 unterscheiden sich darin, dass bei Item I4 vor der *bedingten Anweisung ohne Alternative* eine Anweisung hinzugefügt und die Position der Figur leicht angepasst wurde. Die Daten zeigen, dass der Vergleich der Schwierigkeiten nicht signifikant ist. Eine Erklärung dafür könnte in dem Flüchtigkeitsfehler *bedingte Anweisung ohne Alternative wird als Wiederholung ausgeführt* (Tab. 12.6) und dem Flüchtigkeitsfehler *Übersehen der Anweisung vor der bedingten Anweisung* (27 Vorkommen in I4) liegen. Die Programme der Items I7, I8 und I9 sind alle ähnlich aufgebaut. I7 und I9 wurden so konstruiert, dass eine bestimmte Fehlvorstellung ausgelöst werden soll (siehe Abschn. 12.5), in I8 wurde diese durch den Aufbau des Programms und die Position der Figur bewusst ausgeschlossen. In I8 sollte das *vorzeitige Abbrechen der Wiederholung* explizit keinen Einfluss auf die Schwierigkeit haben. Die Analyse der Fehlvorstellungen zeigt, dass dies wie erwartet umgesetzt werden konnte. In Anbetracht der Daten aus Tab. 12.6 lässt sich schließen, dass die Fehlvorstellung des *vorzeitigen Abbrechens der Wiederholung mit Abbruchbedingung* einen entschiedenen Faktor bei der (in)korrekten Beantwortung von Aufgaben mit Wiederholung mit

Abbruchbedingung darstellt. Eine mögliche Ursache, warum die Vergleiche dennoch signifikant sind, könnte in dem mehrfachen in Erscheinung treten einer zweiten Fehlvorstellung *Wiederholung als bedingte Anweisung* in I8 liegen.

Es zeigt sich, dass leichte Modifizierungen an den Programmen zu keinen statistisch signifikanten Veränderungen der Schwierigkeiten in vier der fünf vorgestellten Fälle führen. Nur in der Paarung I3 und I4 existiert ein signifikanter Unterschied der Schwierigkeiten. Ein möglicher Erklärungsansatz für diese Differenz konnte anhand einer Fehlvorstellung und eines Flüchtigkeitsfehlers gegebenen werden. Das Fehlen eines signifikanten Unterschieds zwischen den Schwierigkeiten von I7, I9 und I8 wirft jedoch eine neue Frage auf. Haben Fehlvorstellungen einen kleineren Einfluss auf die Schwierigkeit eines Items als erwartet? Trotz nicht signifikant unterschiedlicher Schwierigkeiten zwischen I7, I8 und I9 lässt ein detaillierter Blick auf die Traces die Vermutung zu, dass dies nicht der Fall ist.

Zusätzlich war es möglich, Items mit einer vorhersagbaren empirischen Schwierigkeit anhand einer aggregierten Rangfolge zu erzeugen, und es wurden neue Einblicke in die Bearbeitungsprozesse der Testpersonen gewonnen, die für eine automatisierte Generation und Auswertung der Items genutzt werden können.

12.8 Einschränkungen

Die in diesem Beitrag vorgestellte Untersuchung ist an einigen Stellen – zum Beispiel der Auswahl der überprüfenden Items – als prototypisch anzusehen, sodass zunächst weitere Studien folgen müssen, um die Ergebnisse auch im Weiteren zu bestätigen.

Die Bestimmung der Rangfolge durch andere Metriken bzw. mehr Expertinnen und Experten kann andere Ergebnisse aufweisen, jedoch bestätigen unsere empirischen Befunde die von uns angenommene Rangfolge. Bisher nicht untersucht sind dabei aber Einflussfaktoren, wie die Konzentration bzw. die Vorerfahrung im Programmieren, auf die Testdurchführung, das heißt der Vergleich auch komplexerer IRT-Modelle mit dem Rasch-Modell.

12.9 Fazit und Ausblick

Im Rahmen dieses Beitrags wurden zwei Studien behandelt. In der Vorstudie wurden drei Verfahren zum Ermitteln der Schwierigkeit von Testitems untersucht und verglichen. In einem weiteren Schritt wurden die entstandenen Rangfolgen aggregiert, um so eine Rangfolge zu erhalten, die alle drei Verfahren berücksichtigt. Auf dieser Basis wurden neue Items erzeugt, um zu überprüfen, ob die aus der Rangfolge abgeleiteten Regeln als Grundlage der Itemkonstruktion dienen können. Dies konnte in der empirischen Studie bestätigt werden. Darüber hinaus konnte in vier von fünf Fällen gezeigt werden, dass Items, bei denen das Programm als Teil des Items lediglich leichten Modifikationen unterzogen wurde,

sich nicht signifikant in ihren Schwierigkeiten unterscheiden. Für einen der vier Fälle ist das Ergebnis unerwartet. Es wurde erwartet, dass sich die Schwierigkeit des Items I8 signifikant von der Schwierigkeit der Items I7 und I9 unterscheidet, da in diesem die Fehlvorstellung *vorzeitigen Abbrechens der Wiederholung mit Abbruchbedingung* nicht erwartet wurde und auch nicht in den Traces vorzufinden ist. Für die fünf Fälle wurden mögliche Erklärungsansätze aufgezeigt und diskutiert.

Die zukünftige Arbeit sieht vor, anhand der gesammelten Ergebnisse aus beiden Studien einen Generator zu entwickeln, der Items mit einer vorhersagbaren empirischen Schwierigkeit generiert. Mit dem Generator soll es möglich sein, auch gezielt Items zu erzeugen, die eine bestimmte Fehlvorstellung überprüfen. Auch die automatisierte Analyse der unbekannten Items und ein qualitatives Feedback für Lehrkräfte sind ein angedachtes zukünftiges Ziel, damit das Testsystem effektiv in den Programmierunterricht integriert werden kann.

Literatur

Bartholomew, D. J., Steele, F., Moustaki, I., & Galbraith, J. I. (2008). *Analysis of multivariate social science data* (2nd Aufl.). Chapman & Hall/CRC; CRC Press.

Bastian, M., Schneider, Y., & Mühling, A. (2021). Diagnose von Fehlvorstellungen bei der Ablaufverfolgung von Programmen in einem webbasierten Testsystem. In T. Reuter, A. Weber, S. Nitz, & M. Leuchter (Hrsg.), *Problemlösen in digitalen Kontexten* (S. 72–92, Bd. 35). Empirische Pädagogik.

Choi, I.-C., & Moon, Y. (2020). Predicting the difficulty of EFL tests based on corpus linguistic features and expert judgment. *Language Assessment Quarterly, 17*(1), 18–42.

Duran, R., Sorva, J., & Leite, S. (2018). Towards an analysis of program complexity from a cognitive perspective. *ICER'18: Proceedings of the 2018 ACM Conference on International Computering Education Research* (S. 21–30). https://doi.org/10.1145/3230977.3230986.

Europäische Union. (2018). Empfehlung des Rates vom 22. Mai 2018 zu Schlüsselkompetenzen für lebenslanges Lernen.: Amtsblatt der Europäischen Union (C189). https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.C_.2018.189.01.0001.01.ENG. Zugegriffen: 21. Juni 2022.

Fraillon, J., Ainley, J., Schulz, W., Friedman, T., & Duckworth, D. (2020). *Preparing for life in a digital world: IEA International computer and information literacy study 2018 international report.* Springer Nature Switzerland AG.

Frey, A. (2012). Adaptives Testen. In H. Moosbrugger & A. Kelava (Hrsg.), *Testtheorie und Fragebogenkonstruktion* (S. 275–293). Springer.

Hughes, R. T. (1996). Expert judgement as an estimating method. *Information and software technology, 38*(2), 67–75.

Keuning, H., Jeuring, J., & Heeren, B. (2019). A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education (TOCE), 19*(1), 1–43. https://doi.org/10.1145/3231711.

Klauer, K. J. (2014). Formative Leistungsdiagnostik. Historischer Hintergrund und Weiterentwicklung zur Lernverlaufsdiagnostik. In M. Hasselhorn, W. Schneider, & U. Trautwein (Hrsg.), *Lernverlaufsdiagnostik* (S. 1–17, Tests und Trends. 12. Jahrbuch der pädagogisch-psychologischen Diagnostik; Neue Folge). Hogrefe.

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., et al. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin, 36*(4), 119–150.

Lopez, M., Whalley, J., Robbins, P., & Lister, R. (2008). Relationships between reading, tracing and writing skills in introductory programming. *ICER' 08: Proceedings of the Fourth international Workshop on Computing Education Research* (S. 101–112). https://doi.org/10.1145/1404520.1404531.

Mühling, A., Ruf, A., & Hubwieser, P. (2015). Design and first results of a psychometric test for measuring basic programming abilities. *WiPSCE '15: Proceedings of the workshop in primary and secondary computing education* (S. 2–10). https://doi.org/10.1145/2818314.2818320.

Moosbrugger, H., & Kelava, A. (Hrsg.). (2012). *Testtheorie und Fragebogenkonstruktion* (2nd Aufl., S. 68). Springer.

Pea, R. D. (1986). Language-independent conceptual „bugs" in novice programming. *Journal of Educational Computing Research, 2*, 25–36. https://doi.org/10.2190/689T-1R2A-X4W4-29J2.

Perkins, D. N., Hancock, C., Hobbs, R., Martin, F., & Simmons, R. (1986). Conditions of learning in novice programmers. *Journal of Educational Computing Research, 2*, 37–55. https://doi.org/10.2190/GUJT-JCBJ-Q6QU-Q9PL.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., et al. (2009). Scratch: Programming for all. *Communications of the ACM, 52*(11), 60–67.

Rose, S. (2019). *Developing children's computational thinking using programming games*. Hallam University (United Kingdom).

Sorva, J. (2012). *Visual program simulation in introductory programming education*. Zugl.: Espoo, Aalto Univ. School of Science, Diss., (Aalto University publication series Doctoral dissertations, Bd. 61). Aalto Univ. School of Science.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., et al. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*, 127–147. https://doi.org/10.1007/s10956-015-9581-5.