VON MONKIEWITSCH, Till & BÜSCHER, Carina
Köln

Testen und Evaluieren als Computational Thinking Aktivität(en) – Mehr als nur Debugging?!

Theoretischer Hintergrund

Computational Thinking (CT) bezeichnet die Denkprozesse, die zum Formulieren und Lösen von Problemen in einer für Computer verständlichen Form notwendig sind, und besteht aus verschiedenen Aktivitäten wie Abstraktion oder Dekomposition (Shute et al., 2017). Es wird als Voraussetzung zur informierten Teilhabe an unserer zunehmend digitalisierten Gesellschaft gesehen (Shute et al., 2017). Dazu kann und sollte auch der Mathematikunterricht beitragen. Während viele Studien zeigen, dass CT und Mathematik (-unterricht) füreinander lernförderlich sein können, fehlt es an „empirischen Studien, die konkrete Ideen und Praktiken für Lehrende enthalten, die das Lernen von Mathematik und Computational Thinking explizit verbinden“ (Hickmott et al., 2018, S. 65 Übers. d. Verf.). Um diesem Forschungsdesiderat zu genügen, bedarf es genauerer Einblicke in mögliche Lernprozesse (von Schüler*innen), um typische Verläufe und Hürden zu identifizieren (z. B. Büscher, 2024; von Monkiewitsch & Büscher, akzeptiert).

Obwohl Programmieren und CT in der Forschung als unterschiedliche Kompetenzen angesehen werden, wird CT oft durch Programmieren gelehrt, da es seine Anwendung erfordert (Lye & Koh, 2014). Beim Programmieren zeigt sich das Testen im Sinne des Debuggings, also mit dem Ziel der Fehlerbehebung, als besonders relevant für CT (Lye & Koh, 2014). Bereits Carver und Risinger (1987) zeigten aber auch, dass Lernende durch die Vermittlung von systematischem Softwaredebugging befähigt wurden, dabei gewonnene Kompetenzen, zum Auffinden, Identifizieren und Beheben von Fehlern, über das Programmieren hinaus anzuwenden. Shute et al. (2017) sehen diese Fähigkeiten, angewendet, wenn eine Lösung nicht wie vorgesehen funktioniert, als Debugging an – eine wichtige CT-Aktivität. Debugging wird also durch einen Fehler ausgelöst. Büscher (2024) konzeptualisiert das Testen und Evaluieren (TE) als „Testen und gezielte Anpassung“ (S. 4). Hier wird davon ausgegangen, dass auch andere Testgründe (außer Fehler) möglich sind. Um TE-Prozesse genauer erfassen zu können, wurde in von Monkiewitsch und Büscher (akzeptiert), anhand von Prozessen Lernender beim Programmieren geometrischer Figuren, das *Testing and Evaluating Components (TEC) Modell* entwickelt, welches fünf Komponenten rekonstruiert: *Testgrund*, *Test* (Ausführen eines Lösungsplans), *Beobachten* (der Ausführung und/oder des Ergebnisses), *Vergleichen* (mit Erwartung/Spezifikation) und *Konsequenzen*

In: L. Schick, M. Platz & A. Lambert (Hrsg.),
Beiträge zum Mathematikunterricht 2025.
58. Jahrestagung der Gesellschaft für Didaktik der Mathematik. WTM.
https://doi.org/10.37626/GA9783959873307.0

ziehen. Diese können sich zeitlich überlappen, sind nicht trennscharf und sind nicht zwangsläufig in strickt linearer Reihenfolge beobachtbar.

In diesem Beitrag wird TE also allgemeiner gefasst als Debugging, da es auch nicht fehlerbehaftete Situationen betrachtet. Das TEC-Model formuliert den Testgrund als eine TE-Komponente. Offen bleibt aber, welche konkreten Testgründe außer der Fehlerbehebung auftreten können. Diese bieten Einblick in das reflektierte Handeln Lernender, welches Lye und Koh (2014) als Lernvoraussetzung von CT identifizieren. Die Forschungsfrage lautet daher: Welche Testgründe können bei Lernenden rekonstruiert werden?

Methodik

Dieser Beitrag ist eingebettet in eine größere Entwicklungsforschungsstudie (Prediger et al., 2012), die analysiert, *wie* CT und Mathematik(-unterricht) füreinander lernförderlich sein können. Dazu wurden bislang Designexperimente mit 26 Lernenden der Klassen 8–9 (13–15 Jahre), mit Vorkenntnissen in Scratch aus dem Informatikunterricht der 8. Klasse, videographiert. Sie haben paarweise in einem Laborsetting geometrische Figuren, darunter das Haus des Nikolaus (Abb. 1), mithilfe des Turtelcoders (code-your-life.org/turtlecoder) programmiert. Ein Ziel des Projekts ist eine genauere Spezifizierung des Lerngegenstands, wozu insbesondere die Charakterisierung von CT-Aktivitäten gehört. In von Monkiewitsch und Büscher (akzeptiert) wurde das TEC-Modell entwickelt, mit dem die empirisch rekonstruierten aufgetretenen Komponenten des TE (als eine CT-Aktivität) zusammengefasst wurden. Eine dieser Komponenten ist der Testgrund. In diesem Beitrag ist das Ziel wiederum die genauere Ausdifferenzierung möglicher Testgründe.

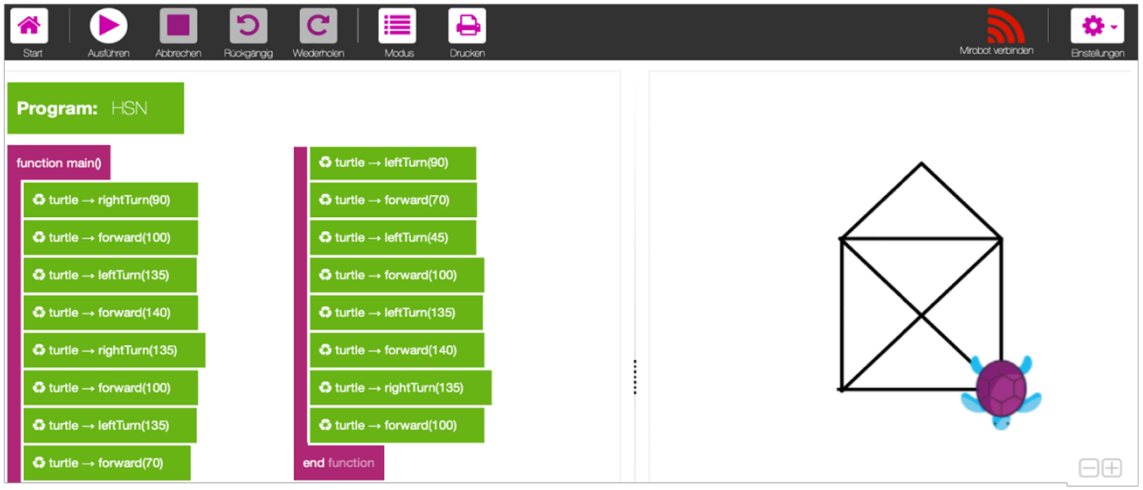

Abb. 1: Nachbildung vom Haus des Nikolaus bei Elena und Esma (Turn 768; siehe auch von Monkiewitsch & Büscher, akzeptiert, S. 3). (© Helliwood media & education)

Im Rahmen einer qualitativen Inhaltsanalyse (Kuckartz & Rädiker, 2022) wurden in den 13 Transkripten obiger Paare die Bearbeitungen vom Haus

des Nikolaus deduktiv mit dem TEC-Modell (von Monkiewitsch & Büscher, akzeptiert) kodiert. Danach wurden die kodierten Abschnitte zum Testgrund in ihrem Kontext analysiert, um verschiedene Testgründe zu inventarisieren.

Empirische Einblicke – vier rekonstruierte Testgründe

Überprüfen einer (mathematischen) Vermutung: Basti und Batoul explorieren, ob ihr Wert für einen Drehparameter passend sein könnte. Sie testen mit 45° eine Vermutung für die passende Winkelgröße, um diese zu überprüfen.

270 Batoul […] und dann … ehm .. ist das jetzt fünfundvierzig?

271 Basti Weiß ich nicht .. lass mal ausprobieren, ich bin ein bisschen unsicher. [Batoul gibt Befehl leftTurn(45) ein & führt Programm aus.]

Überprüfen des Codes: Als Keano und Killian ihre erste Diagonale zeichnen wollen, gibt Keano den Drehwinkel und eine Länge vor. Dann schlägt er einen Test vor um Killian, der sich Keanos Ansatz noch nicht zu eigen gemacht hat, davon zu überzeugen. Im Unterschied zum vorherigen Beispiel geht es hier eher um eine Überprüfung des bisherigen Codes.

552 Keano Ja, vertrau mir. [Killian ändert Befehl leftTurn(45) leftTurn(135) um.]

553 Killian Dann dreht sie sich hier nach, hier machen wir #

554 Keano # Mach erstmal hundertfünfzig. [Killian ergänzt vor penUp den Befehl forward(150).]

555 Keano Oder sollen wir erstmal fahren.

556 Killian Ja [Führt Programm aus]

Entwickeln einer Idee für den Code: Teils fokussieren die Lernenden bereits den nächsten Schritt, indem sie einen Platzhalterwert setzen, um dann den Schritt zu programmieren. Beispielsweise sagt Basti zu Batoul: „Oder wir können ja erstmal weiterlaufen und dann gucken wie es passt, also das dann später ändern“ (Turn 288). Das Ziel ist hier also neue Ideen für den Code zu entwickeln. Dies scheint auch mit einer Dekomposition zusammenzuhängen.

Verstehen des Codes: Felix beschreibt (s)eine Beobachtungsstrategie so: „lass mal die [Turtle] laufen, also ohne Stift dann, oder wenn sie geht das sehen wir auch“ (Finn & Felix, Turn 604). Lässt man die Turtle weiterlaufen, ohne dass sie zeichnet, dann verdeckt ihr Modell kein Teil der Figur mehr. Bei Elena und Esma zeigt Elena Unsicherheit über den gemeinsamen Ansatz, den Esma durch ein Beobachten „wenn das abspielt“ (Turn 598) lösen möchte, denn dann „sieht man ja, was gerade dran ist“ (Turn 601).

598 Esma Warte, lass mal hier, wenn das abspielt gucken.

599 Elena Der Winkel ist zu weit unten.

600 Elena Warte, welcher Winkel ist das?

601 Esma Warte, wenn man abspielt, sieht man ja, was gerade dran ist. [Führt Programm aus.]

Finn und Felix formulieren nach ihrem eben beschriebenen Test direkt Konsequenzen. Dahingegen begründen Elena und Esma diese mit Beobachtungen und Vergleichen. Für beide Paare konnte zwar das Beobachten als Testgrund, nicht aber die Beobachtung als solche mit dem Transkript rekonstruiert werden. Zukünftige Studien können möglicherweise Eye-Tracking einsetzten, um das Beobachten als TE-Komponente genauer zu analysieren.

Fazit

Mithilfe von qualitativen Einblicken in Prozesse von Lernenden konnten wir zeigen, dass TE auch empirisch eine umfassendere CT-Aktivität als Debugging darstellt. Lernende haben mehr Testgründe, als Fehler auszuräumen. Beispielsweise rufen sie ihr Programm auf, um (mathematische) Vermutungen zu prüfen, den Code zu überprüfen, zu verstehen oder ihn weiterzuentwickeln. Diese Testgründe, könnten die Grundlage für Hilfen von Lehrkräften, sowie Lerngelegenheiten zum reflektierten Handeln und (insbesondere ersterer) zum integrierten Lernen von CT und Mathematik bilden. Aktuell bleibt die Forschung durch ihre kleine Stichprobe limitiert, womit das Inventar an rekonstruierten Testgründen keinen Anspruch auf Vollständigkeit hat.

Literatur

Büscher, C. (2024). Differences in Students' Computational Thinking Activities when Designing an Algorithm for Drawing Plane Figures. *International Journal of Science and Mathematics Education*. https://doi.org/10.1007/s10763-024-10465-3

Carver, S. M., & Risinger, S. C. (1987). Improving children's debugging skills. In *Empirical Studies of Programmers: Second Workshop* (S. 147–171). Ablex Publishing

Hickmott, D., Prieto-Rodriguez, E., & Holmes, K. (2018). A Scoping Review of Studies on Computational Thinking in K–12 Mathematics Classrooms. *Digital Experiences in Mathematics Education*, *4*(1), 48–69. https://doi.org/10.1007/s40751-017-0038-8

Kuckartz, U., & Rädiker, S. (2022). *Qualitative Inhaltsanalyse: Methoden, Praxis, Computerunterstützung: Grundlagentexte Methoden* (5. Auflage). Beltz Juventa.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Prediger, S., Link, M., Hinz, R., Hussmann, S., Ralle, B., & Thiele, J. (2012). Lehr-Lernprozesse initiieren und erforschen. *Der mathematische und naturwissenschaftliche Unterricht*, *65*, 452–457.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

von Monkiewitsch, T., & Büscher, C. (akzeptiert). *Investigating students' testing and evaluating as part of computational thinking with the TEC model*. 14th Congress of the European Society for Research in Mathematics Education (CERME14), Bozen-Bolzano, Italy.