bwp@ **Spezial 20** | **November 2023**

Die Förderung von transversalen Kompetenzen in der Berufsbildung

Hrsg. v. **Antje Barabasch & Silke Fischer**

Sabine SEUFERT & Lukas SPIRGI
(Universität St. Gallen)

Programmieren im Zeitalter der generativen KI: eine transversale Kompetenz in der Berufsbildung?

Online:

https://www.bwpat.de/spezial20/seufert_spirgi_spezial20.pdf

www.bwpat.de | ISSN 1618-8543 | *bwp@* 2001–2023

Herausgeber von *bwp@* : Karin Büchter, Franz Gramlinger, H.-Hugo Kremer, Nicole Naeve-Stoß, Karl Wilbers & Lars Windelband

Berufs- und Wirtschaftspädagogik - *online*

SABINE SEUFERT & LUKAS SPIRGI (Universität St. Gallen)

Programmieren im Zeitalter der generativen KI: eine transversale Kompetenz in der Berufsbildung?

Abstract

In diesem Beitrag wird der Frage nachgegangen, inwieweit Programmieren als transversale Kompetenz für die Berufsbildung gefördert werden sollte. Mit den neuesten Entwicklungen im Bereich der Künstlichen Intelligenz (ChatGPT von OpenAI.com) stellt sich die Frage, inwieweit das Erlernen von Programmiersprachen künftig überhaupt noch notwendig sein wird. Um dieser Frage und der Bedeutung von Programmierung in der Berufsbildung nachzugehen, wird zunächst auf den Stand der bildungspolitischen Diskussion eingegangen. Dazu wird das Verständnis von transversalen Kompetenzen, Programmierung als Kompetenz im DigComp Framework sowie das Konzept des Computational Thinking (CT) näher untersucht. CT gilt als eine Kernkompetenz des 21. Jahrhunderts und zielt darauf ab, den Menschen in die Lage zu versetzen, Methoden der Informatik für die Lösung von Problemen in einer Vielzahl an Domänen zu lösen. Neben der Aufarbeitung der bildungspolitischen Diskussion werden anschließend verschiedene Ansätze zum Programmieren, mit denen Berufslernende in Berührung kommen, genauer erörtert: 1) Regelbasiertes Programmieren (insbesondere mit visuellem Coding), 2) Programmieren als Aufsicht im Kontext des maschinellen Lernens und 3) Programmieren in Zusammenarbeit mit generativ vortrainierten Sprachmodellen. In jedem dieser drei Ansätze übernehmen die Berufslernenden eine andere Rolle. Eine Synopse der Überlegungen strukturiert erforderliche Programmierkompetenzen entlang einer KI-Wertschöpfungskette. Damit geben wir einen differenzierten Diskussionsbeitrag auf die Frage, inwieweit noch Programmierkompetenzen gefördert werden sollten, da KI-Systeme dies nun ggf. effizienter bewältigen könnten. Basierend auf unserer Analyse stellen wir vielmehr fest, dass andere Programmierkompetenzen in der Breite erforderlich werden. Berufslernende sollten bereits heute an sog. «Prompting-Skills» (im Sinne von Low-Coding) herangeführt werden, um sie auf zukünftige neue Formen der Zusammenarbeit mit generativen KI-Systemen vorzubereiten.

Programming as a transversal competence in vocational education?

This paper explores the question of the extent to which programming should be promoted as a transversal competence for vocational education and training. With the latest developments in the field of artificial intelligence (ChatGPT by OpenAI.com), the question arises to what extent learning programming languages will be necessary at all in the future. To explore this question and the importance of programming in vocational education and training, the state of the debate on education policy will first be addressed. For this purpose, the understanding of transversal competences, programming as a competence in the DigComp Framework as well as the concept of computational thinking (CT) will be examined in more detail. CT is considered a core competence of the 21st century and aims to enable people to solve computer science methods for solving problems in a variety of domains. In addition to reviewing the educational policy debate, we then discuss in more detail different approaches to programming with which vocational learners come into contact: 1) rule-based programming (especially with visual coding), 2) programming as supervision in the context of machine learning, and 3) programming in collaboration with generatively pre-trained language models. In each of these three approaches,

professional learners take on a different role. A synopsis of our analysis structures required programming competences along an AI value chain. In this way, we provide a differentiated contribution to the discussion on the question of the extent to which programming competences should further be promoted, since AI systems could now manage this more efficiently. Based on our analysis, we rather emphasize that other programming competencies are becoming necessary across the board as transversal competencies. Vocational learners should already be introduced to so-called "prompting skills" (in the sense of low-coding) to prepare them for future new forms of collaboration with generative AI systems.

***Schlüsselwörter**: Transversale Kompetenz, Künstliche Intelligenz, große Sprachmodelle, generative KI, Computational Thinking, Prompting skills, Finetuning*

1 Einleitung

Mit ChatGPT ist nun die Künstliche Intelligenz (KI) im Bildungsbereich, in Schulen und Universitäten sowie in einer breiten Öffentlichkeit angekommen. War die KI bislang eher noch ein sehr abstraktes Thema, das v.a. in Fachkreisen diskutiert wurde, ist die KI nun für alle nutzbar. Die Leistung dieses Tools der KI hat zu der weit verbreiteten Befürchtung geführt, dass Lernende es zum Plagiieren von Aufgaben verwenden, indem sie Antworten auf Aufsatz- oder Prüfungsaufforderungen generieren, anstatt sich die Zeit zu nehmen, ihre eigenen Antworten zu entwickeln. Derzeit beschäftigen sich eher Schulen und Universitäten mit diesem neuen Phänomen. Die Erstellung von Schreibprodukten, generiert durch die KI, mit allen pädagogischen sowie auch juristischen Auswirkungen (v.a. die Frage des geistigen Eigentums) stehen nun im Vordergrund der Debatte. Im kreativen Bereich bei der Erstellung von Kunst, wie z. B. Bildern, Musik durch DALL E (ebenfalls von der Unternehmung OpenAI) findet bereits seit längerem eine öffentliche Debatte darüber statt.

Weniger in der derzeitigen Diskussion steht die Generierung von Programmcode durch das KI Tool ChatGPT. Mit Einführung des DigComp Framework zur Beschreibung relevanter digitaler Kompetenzen im Jahr 2013 für alle Bürgerinnen und Bürger wurde die Programmierung als eine Subdimension der Dimension «Content Creation» eingeführt. Dabei stellte sich bereits die Frage, inwieweit nun Programmierung als eine Art Allgemeinbildung betrachtet werden sollte. Heute stellen sich neue Fragen für die Berufsbildung, da es mit den technologischen Entwicklungen so viel leichter zu sein scheint, Programmieren zu erlernen. Grundsatzfragen stellen sich, wie: „sollen Programmierkenntnisse für Berufslernende gefördert werden, da sie zunehmend wichtige transversale skills darstellen? Oder brauchen wir Programmieren überhaupt nicht mehr lernen, da es der Computer für uns künftig übernehmen wird?“. Die Forschungsfrage soll sich dabei insbesondere auf die neueren Entwicklungen im Zusammenhang mit generativer KI wie ChatGPT beziehen, um erforderliche Kompetenzen, die derzeitig häufig als „prompting skills“ bezeichnet werden, genauer zu analysieren und konzeptionell als transversale Kompetenzen in der Berufsbildung zu verorten. Ziel des Beitrages ist es somit, «Programmieren» als transversale, berufsübergreifende Kompetenz aus der Perspektive für Berufslernende differenzierter zu betrachten und Schlussfolgerungen zu den eben gestellten Fragen, insbesondere zu

den Anforderungen und zur Notwendigkeit von Programmierkompetenzen im Zeitalter der generativen KI zu ziehen.

Dabei folgen wir einem deduktiv-konzeptionellen Vorgehen, das auf der Sichtung und Analyse zentraler Kompetenzmodelle sowie ausgewählter Literatur in den Forschungsfeldern „Transversale Kompetenz“ und „Programmierkompetenzen“ basiert. In einem ersten Schritt erläutern wir in Kapitel 2 den derzeitigen Stand von Programmierung als transversale Kompetenz in der bildungspolitischen Diskussion. In Kapitel 3 gehen wir näher auf unterschiedliche Formen der Programmierung ein, um neben regelbasierten auch Ansätze des maschinellen Lernens zu klären. Auf dieser Grundlage entwickeln wir in Kapitel eine Synopse, indem wir entlang einer KI-Wertschöpfungskette erforderliche Programmierkompetenzen spezifizieren. Kapitel 5 liefert eine Zusammenfassung und einen Ausblick für künftige Forschungsarbeiten.

2 Programmierung als Kompetenz in der bildungspolitischen Diskussion

2.1 Transversale Kompetenzen und Programmierkompetenzen

In der Fachliteratur werden transversale Kompetenz nicht allgemeingültig definiert. (vgl. Scharnhorst/Kaiser 2018, 17). Diverse Publikationen versuchen jedoch die Begriffsvielfallt in Übersichtstabellen zu strukturieren (wie beispielsweise Brewer 2013). Die meisten Definitionsansätze beziehen sich auf das Verständnis von Weinert (2001), der Kompetenz als „die bei Individuen verfügbaren oder durch sie erlernbaren kognitiven Fähigkeiten und Fertigkeiten, um bestimmte Probleme zu lösen, sowie die damit verbundenen motivationalen, volitionalen und sozialen Bereitschaften und Fähigkeiten, um die Problemlösungen in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können“ (Weinert 2001, 27). Daraus wird bereits der hohe Anspruch ersichtlich, dass der Erwerb von eigen- und sozialverantwortlichem Handeln zur Lösung von Problemstellungen im Vordergrund stehen sollte, um flexibel Kompetenzen in unterschiedlichen Situationen anwenden zu können (im Sinne von Problemlösekompetenzen).

Seit den 2000er Jahren findet eine intensive Diskussion über die Bedeutung transversaler Kompetenzen statt (vgl. Scharnhorst 2021). Transversale Kompetenzen können als Querschnittskompetenzen in verschiedenen Kontexten betrachtet werden. Sie können beispielsweise klassische Schulfächer (z. B. Mathematik/Biologie), verschiedene Berufe (z. B. Mechaniker:in/ Schreiner:in) oder auch das Berufs- und Privatleben überspannen (vgl. Scharnhorst/Kaiser 2018, 17). Mittlerweile existieren verschiedene weitverbreitete internationale Referenzrahmen für transversale Kompetenzen, wie beispielsweise der Referenzrahmen fürs lebenslange Lernen der Europäischen Gemeinschaft (2007), die Schlüsselkompetenzen der OECD (2005, 10), das Framework for 21st Century Skillsdie (Battelle for Kids 2019) oder die Publikation der ESCO (Hart et al. 2021) „Towards a structured and consistent terminology on transversal skills and competences“. Die Ähnlichkeit von solchen Konzepten scheint die Notwendigkeit von transversalen Kompetenzen zu belegen (vgl. Scharnhorst/Kaiser 2018, 2). In den verschiedenen Referenzrahmen werden unterschiedliche Begriffe verwendet, wie Schlüsselkompetenzen oder transversal skills (auf Fertigkeiten fokussiert). Neu ist die Idee von transversalen Kompetenzen

nicht. Bereits in den 1970-Jahren forderte Mertens (1974) die Vermittlung von berufs- und funktionsübergreifenden Schlüsselqualifikationen. Der Referenzrahmen „Schlüsselkompetenzen für lebenslanges Lernen“ der EU definiert Schlüsselkompetenzen als situationsgerechte Kombination aus Wissen, Fähigkeiten und Einstellungen (vgl. Europäische Gemeinschaft 2017). Im Zuge der Digitalisierung scheinen transversale Kompetenzen immer wichtiger zu werden (vgl. Scharnhorst 2021). Neben fachlichen und IT-bezogenen Kompetenzen scheinen auch transversale Kompetenzen im Sinne von überfachlichen Kompetenzen, wie Kommunikation, Zusammenarbeit oder Kundenorientierung immer wichtiger zu werden. (vgl. Scharnhorst/Kaiser 2018, 9).

In der beruflichen Aus- und Weiterbildung definiert ESCO den Begriff transversale Kompetenzen: „Transversal skills and competences (TSCs) are learned and proven abilities which are commonly seen as necessary or valuable for effective action in virtually any kind of work, learning or life activity” (Hart et al. 2021, 4). Einen konkreten Vergleich zwischen drei Rahmenkonzepten liefert bereits Scharnhorst (2021, 20), um transversale Kompetenzen näher zu untersuchen. In der nachfolgenden Tabelle ist ein Vergleich dieser Rahmenkonzepte zu finden. Dabei wurde zum einen der Schwerpunkt auf die Analyse von Kompetenzen im Kontext der Digitalisierung gelegt und zum anderen wurde das weit verbreitete ESCO-Modell ergänzt. Darüber hinaus sind die Daten aktualisiert worden, insbesondere das Konzept der Schlüsselkompetenzen der EU (aktualisiert in 2022).

Tabelle 1: Vergleich von Rahmenkonzepten im Hinblick auf transversale Kompetenzen im Kontext der (fortgeschrittenen) Digitalisierung, eigene Darstellung

Rahmen-konzept	**Kompetenzen im Kontext der (fortgeschrittenen) Digitalisierung**	**Erläuterungen und Ausprägungen**
Schlüssel-kompetenzen der OECD (2005, 2019)	Grundkompetenzen und digitale Kompetenz Drei Transformative Kompetenzen (neu ab 2019): Kompetenzen stärken, welche die Künstliche Intelligenz (KI) nicht übernehmen kann	Interaktive Anwendung von Medien und Mitteln Schaffung neuer Werte, Ausgleich von Spannungen und Dilemmata sowie Verantwortungsübernahme
Schlüssel-kompetenzen der EU (2018, 2022)	Grundkompetenzen Transversale Kompetenz	Digitale Kompetenz Bürgerkompetenz (digital citizenship), im Kompetenzbereich 3: Erstellung Digital Content (auch die Programmierung), neu ab 2022 AI Literacy ergänzt (2022)
Partnership for 21st Century Learning (2019)	Grundkompetenzen (core skills) Lern- und Innovationskompetenzen (menschliche Stärken fördern)	Kompetenzen im Umgang mit Informationen, Medien und Technologien 4 K: Kommunikation, Kooperation, Kritisches Denken, Kreativität
ESCO (2021), transversal skills, 3-Ebenen Modell, Life Skills	Core Skills (1. Ebene, innerer Kreis): Working with digital devices and applications 2. Ebene: Self-Management, Physical and Manual, Thinking skills, Social & Communication 3. Ebene: Environmental, Health-Related, Civic, Cultural, Environmental, Financial	Core Skills: Create and edit digital content, Use coding skills Weitere Aspekte, wie z. B. Thinking skills (processing information, ideas and concepts, dealing with problems, thinking creatively and innovatively), social & communication (following ethical code of conduct)

Interessant ist hierbei hervorzuheben, dass die Herausforderung der Künstlichen Intelligenz in derzeit aktualisierten Versionen explizit thematisiert wird. Bei der OECD sind dies die Transformationskompetenzen: „Die Fähigkeit, mit Unsicherheiten zurechtzukommen, neue Haltungen und Werte zu entwickeln sowie produktiv und sinnvoll zu handeln, auch wenn sich Ziele ändern, bleibt vorerst ein Alleinstellungsmerkmal des Menschen. Zum gegenwärtigen Zeitpunkt kann künstliche Intelligenz (KI) mit dem menschlichen Leistungsvermögen, neue Werte zu schaffen, Spannungen auszugleichen oder Verantwortung zu übernehmen, nicht konkurrieren.“ (Bertelsmann Stiftung 2021, 44).

Im Rahmenkonzept der EU, digitale Kompetenzen für Bürgerinnen und Bürger zu definieren, wurde in der letzten Überarbeitung ein separater Anhang mit Fokus KI („Interacting with AI Systems“) ergänzt (vgl. Vuorikari et al. 2022). Von den in diesem Anhang erarbeiteten Aspek-

ten einer AI Literacy wurden 48% Prozent in den bereits existierenden fünf Kompetenzbereichen des DigComp Framewoks übernommen. Um die Reichweite der einzelnen Kompetenzen des DigComp aufzuzeigen, werden zu jeder Kompetenz passende Beispiele im Framework aufgelistet. In allen fünf Kompetenzbereiche werden insgesamt 259 Beispiele aufgelistet. Von all diesen Beispielen besitzen insgesamt 35% einen KI-Bezug und stammen ursprünglich aus dem Anhang „Interacting with AI Systems“. (vgl. Vuorikari et al. 2022). Das DigComp Rahmenkonzept beinhaltet bereits seit der ersten Version im Jahr 2013 Programmieren als Kompetenz für Bürgerinnen und Bürger. Dabei beziehen sich die Kompetenzen sowohl auf den privaten als auch auf den Arbeitsbereich. So sind seit der Version 2.1 von 2017 für alle Kompetenzstufen jeweils Beispiele aus beiden Kontexten aufgeführt. Das Rahmenkonzept hat sich seither im europäischen Raum sehr stark verbreitet und wird auch für den Kontext der Berufsbildung herangezogen. Empirische Studien in der Berufsbildung, um digitale Kompetenzen zu erfassen, greifen häufig auf das DigComp Rahmenkonzept zurück (vgl. hierzu Antonietti/Cattaneo/ Amenduni 2022), Rauseo et al. 2022). Daher erscheint es u.E. besonders geeignet, auf dieses Modell näher einzugehen, um Programmierkompetenzen zu spezifizieren.

2.2 Programmierkompetenzen im Digital Competence Framework for Citizens

Ausgangspunkt bei der Entwicklung des Referenzrahmens bildete das Ziel, politischen Entscheidungsträgern eine evidenzbasierte Grundlage hinsichtlich der Potentiale digitaler Technologien im Bildungswesen und der persönlichen Entwicklung der Bürgerinnen und Bürger zu bieten. Im Konzept werden digitale Kompetenzen definiert, die für einen erfolgreichen digitalen Wandel benötigt werden. Das Framework umfasst insgesamt 21 Kompetenzen und die dazu passenden Anwendungsbeispiele. Die in DigComp Framework aufgenommen Kompetenzen können als Teil der Allgemeinbildung für Bürgerinnen und Bürger angesehen werden. Die definierten Kompetenzen sind demnach sowohl im Privaten wie auch in der Arbeitswelt essenziell. Dies wird insbesondere in den zu jeder Kompetenz vorgestellten Anwendungsbeispielen ersichtlich (vgl. Vuorikari et al. 2022, 8ff.). Gemäß dieser Interpretation entsprechen die im DigComp aufgeführten Kompetenzen auch transversalen Kompetenzen der ESCO-Definition. Sie liegen quer über beide Lebensbereiche.

Im Kompetenzbereich 3 „digital content creation“ wird auf die Programmier-Kompetenzen vertieft eingegangen. (vgl. Vuorikari et al. 2022, 33.). Die Aufnahme dieser Kompetenz ins DigComp Framework zeigt, dass in modernen Arbeitsumgebungen von den Mitarbeitenden (Berufslernenden) auch verlangt wird, digitale Artefakte selbständig zu erschaffen (siehe Kompetenz 3.4). Umgangssprachlich werden diese Programmier-Kompetenzen auch als „Hard Skills“ bezeichnet.

Die erste Fassung des DigComp Framework stammt aus dem Jahr 2013. In dieser Erstfassung wurde Programmierung auf drei Kompetenzstufen ausdifferenziert (vgl. nachfolgende Tabelle). In der Version 2.1 wurden unter anderem die Proficiency Levels des Modells auf acht Stufen erweitert (1 = Foundation bis 8 = Highly Specialised) sowie Beispiele aus der Arbeits- sowie Lernwelt je Kompetenzstufe aufgeführt.

Tabelle 2: Kompetenzstufen der Programmierung nach DigComp (vgl. Vuorikari et al. 2022), eigene Darstellung

<table>
<tr><th colspan="2">DigComp 1.0 (2013)</th><th colspan="2">DigComp 2.1 (2017)</th></tr>
<tr><td colspan="2">To apply settings, programme modification, programme applications, software, devices, to understand the principles of programming, to understand what is behind a programme.</td><td colspan="2">To plan and develop a sequence of understandable instructions for a computing system to solve a given problem or perform a specific task.</td></tr>
<tr><th colspan="4">Proficiency levels</th></tr>
<tr><td rowspan="2">Foundation</td><td rowspan="2">① I can modify some simple function of software and applications (apply basic settings).</td><td rowspan="2">Foundation</td><td>① I can list simple instructions for a computing system to solve a simple problem or perform a simple task.</td></tr>
<tr><td>② I can list simple instructions for a computing system to solve a simple problem or perform a simple task.</td></tr>
<tr><td rowspan="2">Intermediate</td><td rowspan="2">② I can apply several modifications to software and applications (advanced settings, basic programme modifications)</td><td rowspan="2">Intermediate</td><td>③ I can list well-defined and routine instructions for a computing system to solve routine problems or perform routine tasks.</td></tr>
<tr><td>④ I can list instructions for a computing system to solve a given problem or perform a specific task.</td></tr>
<tr><td rowspan="4">Advanced</td><td rowspan="4">③ I can interfere with (open) programmes, modify, change or write source code, I can code and programme in several languages, I understand the systems and functions that are behind programmes.</td><td rowspan="2">Advanced</td><td>⑤ I can operate with instructions for a computing system to solve a different problem or perform different tasks.</td></tr>
<tr><td>⑥ I can determine the most appropriate instructions for a computing system to solve a given problem and perform specific tasks.</td></tr>
<tr><td rowspan="2">Highly specialised</td><td>⑦ I can create solutions to complex problems with limited definition that are related to planning and developing instructions for a computing system and performing a task using a computing system.
I can integrate my knowledge to contribute to professional practice and knowledge and guide others in programming.</td></tr>
<tr><td>⑧ I can create solutions to solve complex problems with many interacting factors that are related to planning and developing instructions for a computing system and performing a task using a computing system.
I can propose new ideas and processes to the field.</td></tr>
</table>

Im Jahr 2022 wurde das aktuelle DigComp Framework 2.2 veröffentlicht. Dabei wurde insbesondere der Kompetenzbereich „Interacting with AI-Systems" mit 77 neuen Ausprägungen ergänzt. (vgl. Vuorikari et al. 2022) Der Bereich 3.4. Programmierung hat dabei keine Ergänzung erfahren. Allerdings sind für die digitale Contenterstellung KI-relevante Aspekte zu berücksichtigen, die folglich auch für die Erstellung von Programmiercode Beachtung finden sollten:

- "3.1 Knows that AI systems can be used to automatically create digital content using existing digital content as its source. Such content may be difficult to distinguish from human creations (Knowledge) (Vuorikari et al. 2022, 33).
- 3.2. Knows how to incorporate AI edited/manipulated digital content in one's own work. This use of AI can be controversial as it raises questions about the role of AI in artworks, and for example, who should be credited (Skills) (Vuorikari et al. 2022, 35).
- 3.5 Considers ethics (including but not limited to human agency and oversight, transparency, non-discrimination, accessibility, and biases and fairness) as one of the core pillars when developing or deploying AI systems" (Vuorikari et al. 2022, 39).

Zum Zeitpunkt der Erstellung des DigComp Frameworks 2.2 im Jahr 2022 war das KI-Tool ChatGPT noch nicht veröffentlicht. Daher könnte aus heutiger Sicht insbesondere 3.2. ergänzt werden um weitere Inhaltsarten, wie KI-generierter Text, Code, etc. Die Analyse der Entwicklung des DigComp Frameworks zeigt auf, dass sich die Verständnisse von Programmierung im Laufe der Jahre verändert haben. Auf diesen Aspekt möchten wir in Kapitel 3 eingehen, um damit auch die veränderten Kompetenzanforderungen differenzierter aufzuzeigen. Im nächsten Abschnitt führen wir zunächst das Konzept des Computational Thinking (im Deutschen häufig übersetzt als informatische Bildung) ein, das als übergreifende Problemlösungskompetenz einen transversalen, berufsübergreifenden Ansatz mit sich bringt und gleichzeitig auch als Brücke zur Programmierung gesehen wird (Israel et al. 2015).

2.3 Computational Thinking als übergreifende Problemlösungskompetenz

Unsere Gesellschaft und unsere Arbeitsweise werden durch enorme technologische Veränderungen geprägt (vgl. Harteis et al. 2020; Ifenthaler et al. 2021; Kirschner/Stoyanov 2020). Der wegweisende Artikel von Wing (2006) bildet den Ausgangspunkt für die aktuelle Diskussion über CT (vgl. Angeli et al. 2016). Wing konzeptualisierte CT als "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (Wing 2006, 33). Damit wird deutlich, dass es sich um eine technologiebasierte Problemlösungskompetenz handelt.

In der CT-Forschung wird die Beziehung zwischen CT und Programmieren häufig thematisiert. Israel et al. (2015) betrachten die Verwendung von Computern zur Modellierung von Ideen und Programmierung als integralen Bestandteil von CT. Buitrago Flórez et al. (2017) sowie Lye und Koh (2014) argumentieren, dass mit Hilfe des Programmierens mehrere Kernaspekte des CT angesprochen werden können. Shute./Sun/Asbell-Clarke (2017) kamen zu dem Schluss, dass es aufgrund ähnlicher zugrunde liegender kognitiver Prozesse eine enge Beziehung zwischen CT und Programmierfähigkeiten gibt. Hsu/Chang/Hung (2018) berichteten auf der

Grundlage ihrer Literaturübersicht, dass das Programmieren weit verbreitet ist, um CT zu unterrichten. Grover et al. (2016) behaupteten, dass das Programmieren einen positiven Einfluss auf die Erfahrungen im CT hat. Scherer/Siddiq/Sánchez Viveros (2019) kamen auf der Grundlage einer Metaanalyse zu dem Schluss, dass CT durch Programmieren unterrichtet werden kann. Die Verwendung professioneller Programmiersprachen wie Java kann jedoch aufgrund der komplexen Syntax für Schülerinnen und Schüler äußerst schwierig sein, und es könnte besser sein, visuelle Programmiersprachen zu verwenden (vgl. Lye/Koh 2014; Repenning 2017). Scratch, das vom Massachusetts Institute of Technology Media Lab (https://scratch.mit.edu) entwickelt wurde, ist eine solche visuelle Programmiersprache, die häufig als Unterrichtsmittel eingesetzt wird (vgl. Hsu/Chang/Hung 2018).

Hinsichtlich der Begründung für die Aufnahme der CT in die obligatorische Schulbildung zeichnen sich nach Bocconi et al. (2016) zwei Haupttrends ab: (1) Entwicklung von CT-Fähigkeiten bei Kindern und Jugendlichen, um sie in die Lage zu versetzen, anders zu denken, sich über eine Vielzahl von Medien auszudrücken, reale Probleme zu lösen und alltägliche Probleme aus einer anderen Perspektive zu analysieren; und (2) Förderung der CT aus ökonomischen Gründen, um sich auf eine künftige Beschäftigung vorzubereiten (vgl. Bocconi et al. 2016).

Seit dem Durchgang im Jahr 2018 ist CT auch Gegenstand der International Computer and Information Literacy Study (ICILS). Diese Studie auf Sekundarstufe I wurde im Jahr 2013 erstmalig durchgeführt. Neben CT werden Information Literacy und Computer Literacy abgeprüft (Fraillon et al. 2018). Den Autoren der Studie nach handelt es sich bei CT um eine weitere, überfachliche Kompetenz im Sinne einer vertieften technologischen Literacy. Über die Ausdifferenzierung von CT in Kompetenzfacetten herrscht noch kein Konsens. Das Rahmenkonzept von Brennan und Resnick (2012) dient oft als theoretische Grundlage für CT. Es umfasst drei Dimensionen (vgl. Brennan/Resnick 2012, 3ff.):

1. "*Computational concepts*" sind Denkmuster: 1. Logik und logisches Denken, 2. Algorithmen und algorithmisches Denken, 3. Muster und Musterkennung, 4. Abstraktion und Generalisierung, 5. Evaluation und 6. Automation (vgl. Grover/Pea 2017).
2. "*Computational practices*" sind Prozesse für Problemlösungen: 1. Problemzerlegung, 2. computational Artefakte entwickeln, 3. Testen und «debuggen», 4. Iterative Verfeinerung (inkrementelle Entwicklung) sowie 5. Kollaboration und Kreativität (als Teil eines breiteren Verständnisses von Schlüsselkompetenzen des 21. Jahrhunderts) (vgl. Grover/Pea 2017).
3. "*Computerperspektiven*" sind Verschiebungen der Perspektiven, der Beziehung zu anderen und der digitalen Welt (shift in world view); Beispielsweise sollten junge Menschen sich mit Fragen der Technologie auseinandersetzen: „I can (use computation to) ask questions to make sense of (computational things in) the world“ (Brennan/Resnick 2012, 11).

Die grundlegende Idee von CT ist es daher, dass Lernenden nicht nur technikkundig sind (im Sinne von „digital literacy“), sondern rechnergestützte Werkzeuge, computational tools (zu-

nehmend auch KI-Tools) zur Lösung von Problemen in einer veränderten Denkweise einsetzen können (vgl. Grover/Pea 2013). Repenning (2019) sieht in CT deshalb eine zentrale Kernkompetenz, da damit eine neue Interdisziplinarität gefördert werden kann, Zusammenhänge der Disziplinen verstehen und nutzen zu können. In diesem Zusammenhang versteht Repenning (2018) Interdisziplinarität als eine kombinierte und oft problemlösungsorientierte Nutzung von Ansätzen, Denkweisen und Methoden aus verschiedenen Fachgebieten. Interdisziplinarität benötigt einen gut funktionierenden Verständigungsprozess, um eine gemeinsame Sprache und ein gemeinsames Verständnis zwischen den verschiedenen Interessensgruppen zu ermöglichen (vgl. Yadav et al. 2018). CT könnte daher künftig wie eine «Verständigungssprache» zwischen den unterschiedlichen Disziplinen wirken. CT-Praktiken könnten dabei unterstützen, metakognitive Prozesse zu unterstützen, wie die Zusammenstellung und Beispiele in der nachfolgenden Tabelle aufzeigen:

Tabelle 3: CT-Praktiken, metakognitive Prozesse und Beispiele (Yadav et al. 2022, 409), eigene Darstellung

Name of the CT Practice	Associated Metacognitive Processes	Examples
Abstraction: Focusing on the most relevant and essential details of the problem that needs to be solved.	• Defining a problem.	• Deciding on how to represent a phenomenon through a computational model.
	• Selecting the relevant elements from the problem to be addressed.	• Picking representations of the world using maps based on what features (country boundaries, topography, etc.) need to be focused on. Depending on the features, the world map looks different.
	• Attending critical features of a problem.	
Decomposition: Simplifying complex tasks by breaking them down into smaller parts	• Decomposing a task into sub-problems that are well-structured to decrease task complexity.	• Using decomposition to find the area of a rectangle in elementary mathematics
		• In Scratch, writing a complex program by focusing on individual Sprites and writing associated code
Algorithmic thinking: Designing step-by-step solutions to a problem.	• Planning how to proceed.	• Making dinner for guests, including deciding what to cook, what steps to follow while cooking, and when to cook so that food is ready on time.
	• Identifying steps needed to solve the problem.	• Writing a program that creates a polygon based on the number of sides input by the user.
	• Executing the steps serially or in parallel.	
Debugging: Finding and fixing errors.	• Monitoring the solution.	• When a lamp stops working, using troubleshooting to figure out the cause of the problem and fix it: whether it is electricity, it is plugged in, or it is the bulb that burned out.
	• Assessing the solution.	
	• Trying new strategies when the former one doesn't work.	• Fixing errors in the model or a program that does not produce the desired output.

CT-Forschergruppen (z. B. Rode et al. 2015; Pollak/Ebner 2019) plädieren in neueren Ansätzen auch dafür, die «Maker Bewegung» und damit verbunden eine neu entstandene «DIY-(Do-it-yourself)-Kultur» zu nutzen, um Lernende unterschiedlicher Herkunft und Altersgruppen mit-

einander zu interagieren und voneinander lernen zu lassen. Darüber hinaus unterstützt die Makerbewegung, eher abstrakte Lerngegenstände (wie z. B. Internet der Dinge) durch direktes Tun und haptische Lernvorgänge besser zu verstehen (vgl. Wey-Han 2011). Das Maker-Movement ist in zweierlei mit gesellschaftlichen Herausforderungen verbunden: Einerseits ist zu beobachten, dass Kinder und Jugendliche heutzutage immer weniger basteln und tüfteln aufgrund der Digitalisierung (z. B. Games, eSport) und veränderten Freizeitaktivitäten (z. B. weniger gemeinsam reparieren, vgl. Ratto 2011). Dadurch besteht die Gefahr, dass feinmotorische Fähigkeiten und handwerkliches Geschick immer mehr verkümmern. Andererseits entwickeln sich immer mehr DIY-(„do it yourself“)-Konzepte in unserer Gesellschaft und Kunden werden z. B. immer mehr in Co-Creation Prozesse eingebunden, um Produkte und Dienstleistungen gemeinsam zu verbessern (Seufert 2017). Um das volle Potential der Makerbewegung auszuschöpfen, kann CT also zu Computational Making (CM) erweitert werden (z. B. Rode et al. 2015; Grover/Pea 2017). Als Taxonomie in diesem Bereich schlagen die Autoren fünf Schlüsselfaktoren zur Erweiterung des CT-Ansatzes vor: Ästhetik, Kreativität, Konstruktion, Visualisierung multipler Repräsentationen und das Verständnis für Materialien, um damit vor allem auch die individuelle Kreativität und die Zusammenarbeit in heterogenen Teams anzusprechen. In derartigen Projektarbeiten könnten fachliche Kompetenzen, wie z. B. die nachhaltige Entwicklung von Produkten zur Erhaltung der Umweltressourcen, intensiv bearbeitet werden. Gleichermaßen könnten methodische Kompetenzen zur kreativen und analytischen Problemlösung gefördert werden. CT bzw. auch CM könnten daher besonders für die Berufsbildung interessante Konzeptionen darstellen, um Lernenden die Möglichkeit zu bieten, generische, transversale Kompetenzen im Umgang mit Programmierung zu erlernen.

Nach dieser bildungspolitischen Betrachtung möchten wir im anschließenden Kapitel auf die unterschiedlichen Ansätze der Programmierung eingehen, um damit auch die verschiedenen Herausforderungen und Anforderungen an die jeweiligen Kompetenzen zu klären.

3 Unterschiedliche Betrachtungsweisen der Programmierung

Die Art und Weise wie programmiert wird, hat sich im Laufe der Zeit weiterentwickelt. In der heutigen Arbeitswelt könnten Berufslernende mit drei unterschiedlichen Verständnissen von Programmierung in Kontakt geraten, welche auch jeweils unterschiedlichen Kompetenzen verlangen. Diese Ansätze werden zunächst differenziert betrachtet und insbesondere auf die mögliche Rolle von Fachkräften in der Berufsbildung eingegangen.

3.1 Regelbasiertes Programmieren

Das regelbasierte Programmieren entspricht der klassischen Programmierung. Ursprünglich definiert ein Entwickler:in explizite Anweisungen in verschiedenen Programmiersprachen wie Python, Java, JavaScript, C++. Das vom Entwickler:in erstellte Computerprogramm liefert in Kombination mit dem zur Verfügung gestellten Datensatz den gewünschten Output (vgl. Delipetrev et al. 2020, 11). Die benötigten Kompetenzen, um einen regelbasierten Computercode zu erstellen, werden im DigComp Framework unter der Kompetenz 3.4 „Programming“ aufgearbeitet. Die Programmier-Kompetenz wird wie folgt definiert: “To plan and develop a

sequence of understandable instructions for a computing system to solve a given problem or perform a specific task" (Vuorikari et al. 2022, 33). Ein Anwendungsbeispiel dieser Art des Programmierens ist, wenn Berufslernende ein Computerprogramm erstellen, das ein interner Arbeitsprozess automatisiert.

Die Art und Weise, wie Berufslernende solche Aufgaben lösen können, hat sich im Laufe der Zeit verändert. Bereits im Jahr 2014 wurde der Begriff 'Low Code' eingeführt (vgl. Ruscio et al. 2022). Die klassische Herangehensweise des Programmierens kann mit verschiedenen unterstützenden Technologien für Nicht-Expert:innen einfacher zugänglich gemacht werden. Low-Code-Plattformen (LCDP) besitzen eine grafische Benutzeroberfläche, so dass Entwickler:innen weniger Zeit für die Generierung des eigentlichen Computercodes verwenden müssen und mehr Zeit in die Ästhetik und Funktionalität der Anwendung investieren können (vgl. Waszkowski 2019, 377). App-Applikationen ermöglichen die Entwicklung von Anwendungen mit Hilfe von einfachen Workflows, Datenvisualisierung und weiteren Tools (vgl. nachfolgende Abbildung). Die Verwendung von Low-Code-Plattformen für die Automatisierung von Geschäftsprozessen hat das Potenzial, die Kosten und den Zeitaufwand für die Implementierung, Entwicklung und Pflege von Prozessen erheblich zu reduzieren (vgl. Waszkowski 2019, 377).

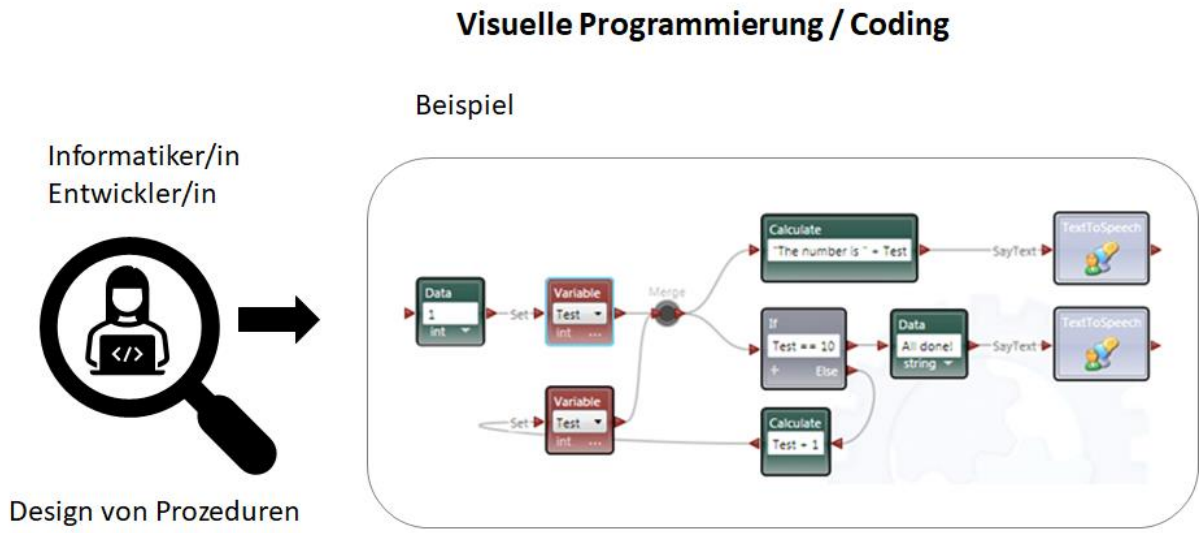

Abbildung 1: Regelbasiertes Programmieren, eigene Darstellung (Bildquelle: https://www.usabilityfirst.com/glossary/visual-programming-language/index.html)

Wie aufgezeigt, ist aufgrund neuer Technologien die Kenntnis von traditionellen Programmiersprachen (z. B. Python, Java) für das regelbasierte Programmieren nicht mehr zwingend notwendig. Weiterhin zentral bleibt jedoch das Verstehen der grundsätzlichen Systematik des regelbasierten Programmierens, wonach ein Entwickler:in dem Computer strukturierte Anweisungen erteilen muss, damit dieser einen Output generieren kann (vgl. Delipetrev et al. 2020, 12). Durch die technologische Entwicklung könnte damit die Zugangsschwelle zum Programmieren für Berufslernende bereits gesenkt worden sein.

3.2 Programmierung mit Supervision im Kontext des maschinellen Lernens

Im Gegensatz zum klassischen, regelbasierten Ansatz wird ein System mit maschinellem Lernen eher trainiert als programmiert (vgl. Delipetrev et al. 2020, 11). Machine Learning (ML) ist ein statistischer Ansatz, der es Computern ermöglicht, aus Daten zu lernen (vgl. LeCun/Bengio/Hinton, 2015). Solche KI-Anwendungen sind für die Berufsbildung besonders wichtig, da sie tiefgreifende Veränderungen bei der Beschäftigung und ihren Arbeitsaufgaben bedeuten. Es wird auch zu veränderten Aufgaben und Rollen innerhalb von bestehenden Arbeitsplätzen kommen (vgl. Makridakis 2017; Seufert 2017). Eine mögliche neue Aufgabe ist, dass Berufslernende in Zusammenarbeit mit Informatiker:innen mithelfen spezifische ML-Anwendungen zu entwickeln. Die Berufslernenden helfen beim Trainieren der KI-Anwendung in dem sie das Labeling des Trainingsdatensatzes übernehmen (supervised learning). So bringen sie ihre spezifische Fachexpertise in die Entwicklung des KI-Systems ein (vgl. nachfolgende Abbildung). Informatiker:innen entwickeln den ML-Algorithmus des Sprachmodells und die Fachkräfte liefern den gelabelten Trainingsdatensatz mit ihrer Fachexpertise. Bei der Entwicklung von KI-basierten Systemen (interdisziplinären ML-Teams) mitzuwirken, ist daher in allen Berufsfeldern ein steigendes Zukunftsfeld.

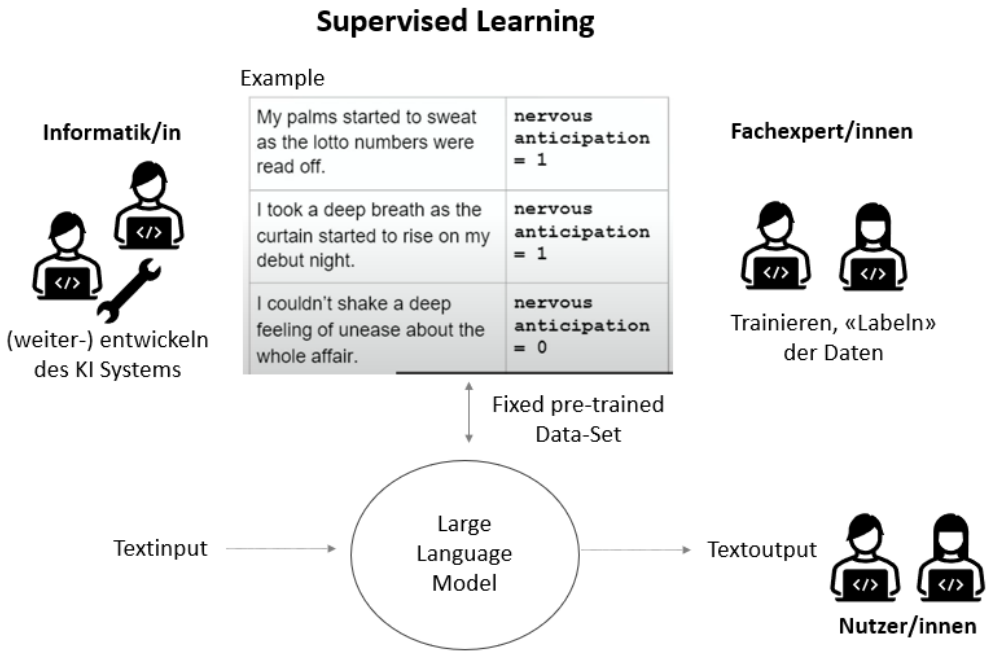

Abbildung 2: Trainieren eines LLM, eigene Darstellung (Bildquelle: https://www.youtube.com/watch?v=-lnHHWRCDGk)

Damit Berufslernende gewinnbringend in solchen ML-Teams mitarbeiten könnten, müssen sie neue Kompetenzen entwickeln. Beispielsweise benötigen die Berufslernenden ein grundlegendes Verständnis für die neuen Technologien damit sie selbstbewusst, kritisch und sicher mit neuen KI-Anwendungen arbeiten können (vgl. Vuorikari/Holmes 2022, 77). Außerdem müssen die Berufslernenden eine gut ausgebildete Daten-Affinität entwickeln. Die dem ML-Algorithmus zugrunde liegenden Daten sind entscheidend für das erfolgreiche Erstellen von Vorhersagemodellen und bilden die Grundlage für ML. In den Input-Daten können Befangen-

heiten oder Einseitigkeiten vorhanden sein, welche den Algorithmus, respektive das erstelle Vorhersagemodell beeinflussen. Wenn verzerrte Datensätze zum Training von KI-Systemen genutzt werden, führt dies zwangsläufig auch zu verzerrtem Output der KI-Modelle (vgl. Caliskan/Bryson/Narayanan 2017).

3.3 Programmierung mittels vortrainierten Sprachmodellen

Ein weiterer großer Schritt in der Entwicklung von KI-Anwendungen könnten die sogenannten „Foundation Models“ (generative, vor-trainierter Modelle) sein (vgl. Potts 2023). Solche Modelle werden auf Basis von riesigen Datenmengen trainiert und können für unterschiedliche spezifische KI-Aufgaben adaptiert („fine-tuning“, vgl. Bommasani et al. 2021, 4) werden. Die Trainingsdaten von Foundation Models stammen aus diversen Quellen und werden zentralisiert, dies führt dazu, dass nicht für jeden neuen Task ein neues Modell trainiert werden muss. So wird die Skalierbarkeit, die Wirtschaftlichkeit und die Anwenderfreundlichkeit von KI-Anwendungen verbessert (vgl. Bommasani et al. 2021).

Im Zusammenhang mit der Kompetenz ‘Programmieren’ sind primär generativ vor-trainierte Sprachmodelle interessant, denn diese KI-Modelle sind in der Lage Computercode selbständig zu generieren (vgl. OpenAI 2023). Solche Sprachmodele werden als Large Language Models (LLMs) bezeichnet und basieren auf der Technologie von Natural Language Processing (NLP), welche Daten in Form von natürlicher Sprache verarbeitet (Potts 2023). Ein aktuelles Beispiel eines generativen LLM ist ChatGPT. Große Sprachmodelle (LLMs) könnten die Art und Weise des Programmierens verändern. Mit Hilfe von einem KI-Sprachmodell und einem geschickten Prompt-Engineering können Fehler in bereits erstelltem Computercode gesucht werden oder ein neuer Computercode generiert werden (vgl. OpenAI 2023). Diese Technik könnten Berufslernende nutzen, um Computer-Programme zu erstellen, welche sie in ihren Arbeitstag verwenden. Einigen Berufslernenden könnte es erleichtert werden, ein höheres Proficiency Level der Programmier-Kompetenzen des DigComp Frameworks zu erreichen (vgl. Vuorikari et al. 2022, 33). Das eigentliche Coden wird vom Sprachmodell übernommen und die Berufslernenden können sich auf das konzeptionelle Design der Problemlösung, Funktionen sowie auf die Qualitätsüberprüfung des zu erstellenden Programmes konzentrieren.

Im Zusammenhang solcher LLMs entwickelt sich aktuell ein neues Lernparadigma für ML. Beim „in-context learning“ wird ein Sprachmodell trainiert, indem der Nutzer:in neben der eigentlichen Aufgabe auch Beispiellösungen vorschlägt (vgl. Mc Cann et al. 2018; Brown et al. 2020). Die Sprachmodelle müssen also nicht mehr aus dem Stand eine Antwort generieren (Zero-Shot-Strategie), sondern lernen aus einer (One-Shot-Strategie) oder mehreren (Few-Shot-Strategie) Beispiellösungen des Nutzer:in (vgl. Finkbeiner/Schmitt 2021). Mit diesem „in-context learning“ könnten Berufslernende durch die Nutzung des Sprachmodells ihre Fachexpertise einbringen und das KI-Model weiter trainieren. Das Modell wird also im Kontext trainiert und nicht nur auf Basis von Datensätzen (Potts 2023).

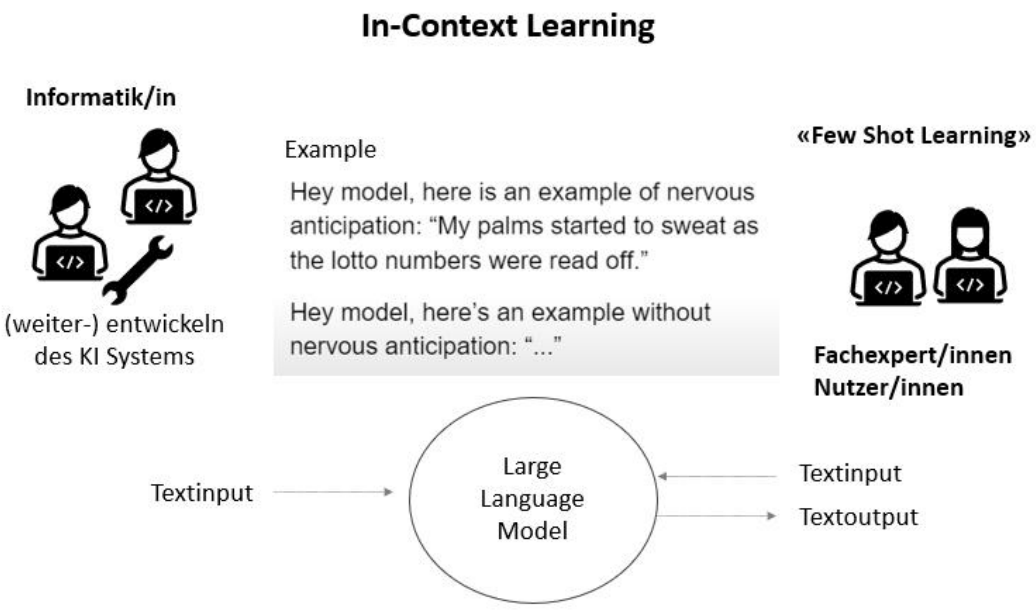

Abbildung 3: In-Context Learning, eigene Darstellung, (Bildquelle: https://www.youtube.com/watch?v=-lnHHWRCDGk)

Das Sprachmodell GPT-3 sowie auch die neuere Version GPT-4 wurde auf einer großen Menge an Text aus dem offenen Internet vortrainiert. Wenn es einen Prompt mit nur wenigen Beispielen erhält, kann es oft intuitiv erkennen, welche Aufgabe Benutzer:innen ausführen möchten und eine plausible Vervollständigung generieren. Dies wird dann eben als „Few-shot Learning" bezeichnet.

Generell zeigen die Entwicklungen rundum ChatGPT, dass es dabei um eine neue Kategorie von KI und maschinellem Lernen darstellt, die als generative KI auf Basis von großen Sprachmodellen bezeichent werden kann. Laut Lim et al. (2023, S. 2) kann generative KI als eine Technologie definiert werden, die deep learning nutzt, um menschenähnliche Inhalte (z. B. Bilder, Wörter) in Reaktion auf komplexe und vielfältige Aufforderungen (z. B. Sprachen, Anweisungen, Fragen) zu generieren. Die von ChatGPT generierten Antworten sind stark abhängig von den Eingaben, die als "Prompts" bezeichnet werden, die es in Bezug auf Menge und Qualität erhält (Lim et al. 2023). Wenn ChatGPT die richtigen Anweisungen erhält, kann es eine detailliertere Antwort liefern. Daher haben viele Forscher sogenannte "Prompting Guides" entwickelt (Gimpel et al. 2023; Herft et al. 2023). Derzeit sind die "Prompting Skills" für die kompetente Nutzung von ChatGPT neu in Diskussion, die auch als „low coding skills" betrachtet werden können (Herft et al. 2023). Von daher können Programmierkompetenzen in dieser Form für die Anwendung generativer KI wie ChatGPT als neue Allgemeinbildung und transversale Kompetenz betrachtet werden.

Für die Entwicklung von Webservices, die auf diesen großen Sprachmodellen aufbauen, sind darüber hinaus anspruchsvollere Kompetenzen im Bereich des Prompt Engineerings und Finetuning notwendig, wie nachfolgende Abbildung zunächst aufzeigt:

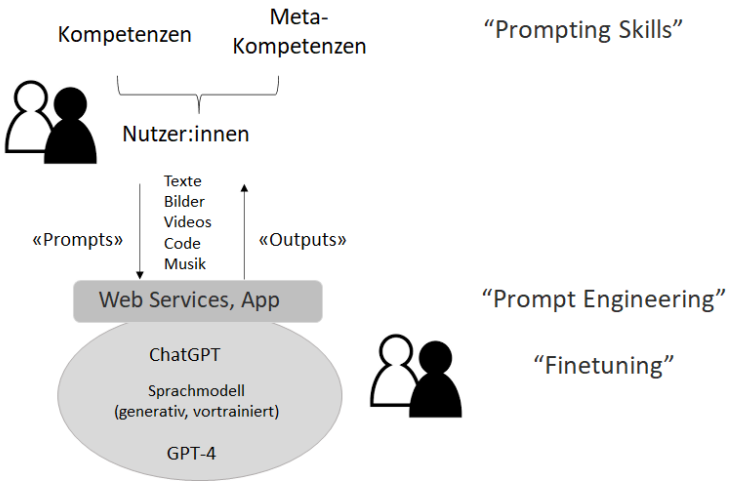

Abbildung 4: Prompting Skills, Prompt Engineering und Finetuning, eigene Darstellung

«Prompt-Engineering» bezieht sich auf den Prozess der Gestaltung und Optimierung der Prompts, die zur Interaktion mit einem Sprachmodell verwendet werden. Durch geschicktes Prompt-Engineering kann die Ausgabe des Sprachmodells gesteuert und präziser und relevanter für eine bestimmte Aufgabe oder ein bestimmtes Themengebiet gestaltet werden. Dies umfasst die Auswahl der am besten geeigneten Prompts für eine bestimmte Aufgabe, deren Verfeinerung zur Verbesserung ihrer Effektivität und die Überprüfung, ob sie die gewünschte Ausgabe generieren. Prompt-Engineering kann verschiedene Techniken umfassen, einschließlich der Verwendung von Vorlagen, der Auswahl relevanter Schlüsselwörter und der Anpassung von Prompt-Länge und -Struktur. Geschicktes Prompt-Engineering kann dazu beitragen, den Kontext sowie den Stil des Ausgabetextes einzuschränken. Dies reduziert die Wahrscheinlichkeit, unerwünschte Inhalte zu erhalten.

Zusammenfassend lässt sich sagen, dass «Prompting» das Eingeben eines spezifischen Prompts in ein Sprachmodell bezeichnet, um eine bestimmte Ausgabe zu generieren, während «Prompt-Engineering» die Gestaltung und Optimierung von Prompts umfasst, um das Sprachmodell für eine bestimmte Aufgabe oder ein bestimmtes Themengebiet effektiver zu gestalten.

Beim «Finetuning» handelt es sich um eine Machine-Learning-Technik, bei der ein bereits trainiertes Modell für eine neue Aufgabe oder einen neuen Bereich weiter trainiert wird, um die Leistung bei dieser spezifischen und wiederkehrenden Aufgabe zu verbessern. Beim "Finetuning" handelt es sich um einen Prozess des weiteren Trainierens eines bereits vortrainierten Modells auf spezifische Aufgaben oder Daten. Es wird verwendet, um die Leistung des Modells zu verbessern und es an bestimmte Anforderungen oder Domänen anzupassen. Beim Finetuning werden in der Regel die Gewichtungen und Parameter des vortrainierten Modells angepasst, um die Vorhersagegenauigkeit für die spezifische Aufgabe oder Datenmenge zu optimieren. Dies ermöglicht es dem Modell, besser auf neue oder spezialisierte Aufgaben zu reagieren und eine höhere Leistungsfähigkeit zu erreichen, als wenn es von Grund auf neu trainiert würde. Häufig wird das Modell dafür auf einem kleineren Datensatz trainiert, der Beispiele dieser spezifischen Aufgabe enthält. Durch das Training des Modells auf den neuen Daten kann das

Modell Muster und Beziehungen spezifisch für die neue Aufgabe erkennen und seine Leistung bezüglich dieser spezifischen Aufgabe verbessern.

Beim Fine-tuning von großen Sprachmodellen wird das Modell für eine spezifische Aufgabe wie z. B. Text-Klassifizierung, Frage-Antworten oder Text-Generierung weiter trainiert, indem es auf einer kleineren Menge an spezifischen Trainingsdaten angepasst wird. Fine-Tuning ermöglicht es somit, die Leistungsfähigkeit des Modells weiter zu steigern. Fine-Tuning verbessert das «Few-shot-Learning», indem es auf mehr spezifische Beispiele trainiert ist als in einen Prompt passen, so dass man bessere Ergebnisse auf einer Vielzahl von Aufgaben erzielen kann. Sobald ein Modell feinabgestimmt wurde, müssen keine Beispiele mehr im Prompt bereitgestellt werden («Zero-shot Learning»). Dies spart Zeit sowie Token-Kosten und ermöglicht Anfragen mit niedrigerer Latenzzeit (schnellere Antwortzeit).

Aktuell gibt es sicherlich noch große Herausforderungen und Limitationen dieser LLMs. Eine zentrale Schwierigkeit ist die Abschwächung der aktuell noch bestehenden Verzerrungen in den Basismodellen. Außerdem muss die Robustheit der KI-Modelle verbessert werden (vgl. Yang et al. 2018). Die Berufslernenden müssten auch die Schwächen und Limitationen von Foundation Models kennen (z. B. das Problem der Datenhalluzination (vgl. Handschuh 2023). Das Verhalten von solchen generativ vor-trainierten Modellen kann teilweise unvorhersehbar sein (vgl. Bommasani et al. 2021). Dieses Problem wird im DigComp Framework explizt angesprochen: “Aware that AI algorithms work in ways that are usually not visible or easily understood by users. This is often referred to as “black box” decision-making as it may be impossible to trace back how and why an algorithm makes specific suggestions or predictions” (Vuorikari et al. 2022, 86). Das angesprochene Black-Box-Problem beschreibt das Phänomen, dass für Nutzer:innen einer KI-Anwendung oft nicht mehr erklärbar und nachvollziehbar ist, wie ein bestimmter Output des Systems zustande gekommen ist. Selbst für die Entwickler:innen ist die Entscheidungsfindung des Algorithmus oftmals nicht im Detail nachvollziehbar. (vgl. Christen et al. 2020, 56ff.). Außerdem muss bei der Entwicklung solcher KI-Anwendungen immer auf die Einhaltung von ethischen Grundsätzen geachtet werden. Im DigComp Framwork wird insbesondere die benötigten Einstellungen für eine Zusammenarbeit mit KI-Anwendungen konkretisiert, wie im Abschnitt 2.2 aufgeführt. Dies stellt eine neue Kompetenzanforderung für die Programmierung mit KI-Tools dar, ethische Überlegungen anstellen zu können (vgl. Vuorikari et al. 2022, 33).

Als Zwischenfazit ist festzuhalten, dass in vielen Berufsfeldern und Branchen Programmierkompetenzen gefragt sein könnten. Zum einen da klassische Ansätze des regelbasierten Programmierens für die Automatisierung von Routinetätigkeiten künftig zunehmend wird. Zum anderen sind neue Rollen bei der Entwicklung von KI-Systemen erforderlich, welche sich insbesondere auf das Training von Daten mit entsprechender Fachexpertise bezieht. Sowohl beim Supervised Learning als auch beim in-context-Learning bzw. Prompt Engineering grosser Sprachmodelle sind neue Kompetenzanforderungen erforderlich, welche sich auf Datenqualität sowie auch auf ethische Anforderungen beziehen. Aufgrund dieser dynamischen Entwicklungen soll im nächsten Kapitel näher darauf eingegangen, um eine Synopse entlang einer KI-Wertschöpfungskette zu entwickeln.

4 Synopse: Programmierkompetenzen im Zeitalter der KI

Als Synopse der vorgegangenen Überlegungen soll als Strukturierungselement eine KI-Wertschöpfungskette dienen, um die unterschiedlichen Anforderungen an Programmierkompetenzen zu klären. Die KI-Wertschöpfungskette auf der Basis großer Sprachmodelle kann wie folgt anhand der folgenden Entitäten strukturiert werden (vgl. hierzu Hacker/Engel/Mauer 2023):

- Entwickler:innen, die das Modell ursprünglich erstellt und (vor-)trainiert. Reale Beispiele wären OpenAI, Stability oder Google.
- Serviceanbieter, die das Modell für einen spezifischen Anwendungsfall feinabstimmt („Finetuning“) und damit Webservices bereitstellt.
- Professioneller Benutzer:innen, die die KI-Ausgabe für professionelle Zwecke verwendet, wie z. B. ein gewinnorientiertes oder gemeinnütziges Unternehmen, eine NGO, eine Verwaltungsbehörde, ein Gericht oder der Gesetzgeber.
- Nicht-professioneller Benutzer:innen, die die KI-Ausgabe für nicht-professionelle Zwecke verwendet (wie in den EU-Verbraucherrechten definiert).
- Empfänger:innen, die das vom Benutzer angebotene Produkt konsumiert. Typischerweise wird es sich um einen Verbraucher handeln, aber es könnte auch ein Unternehmen, eine NGO, eine Verwaltungsbehörde, ein Gericht oder der Gesetzgeber sein. Was den Empfänger in all diesen Fällen auszeichnet, ist, dass er sich am passiven, empfangenden Ende der Pipeline befindet. Hierbei geht es vor allem darum, sich auch über die eigenen Rechte im Sinne einer KI-Literacy (vgl. DigComp Framework) bewusst zu sein.

„Prompting Skills“ und “Prompt Engineering” könnten sich daher in sehr vielen Berufsfeldern als neue Kompetenzanforderungen ergeben. Für heutige Berufslernende könnten diese Kompetenzen künftige Tätigkeitsfelder darstellen, um mit ihrer Fachexpertise in interdisziplinären Teams zusammen zu arbeiten.

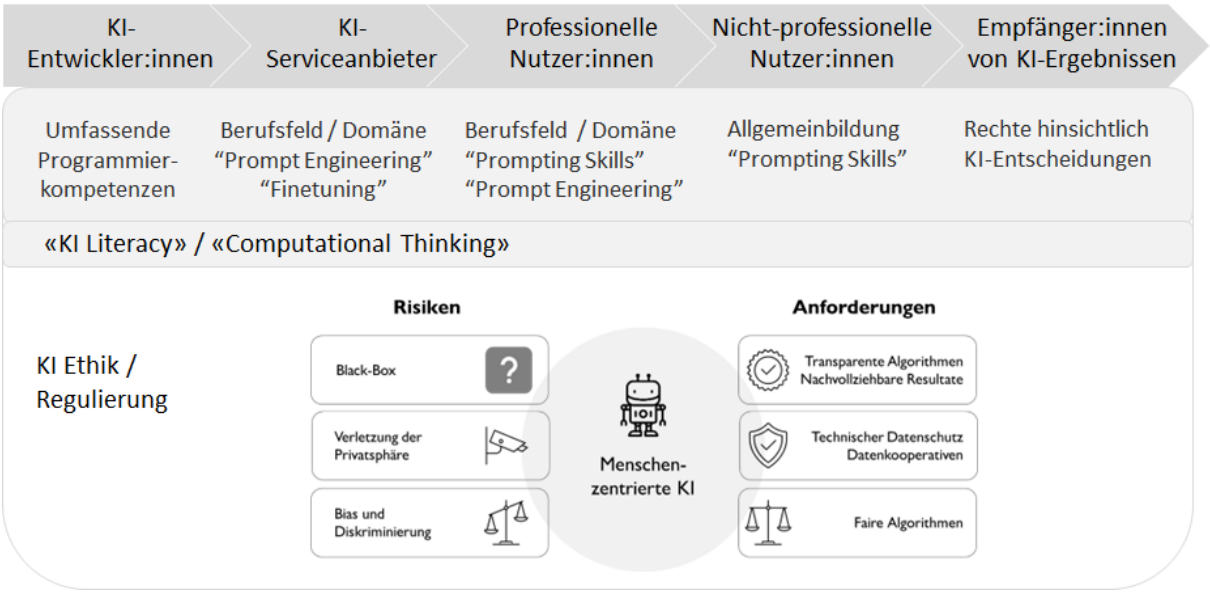

Abbildung 5: Programmierkompetenzen entlang der KI-Wertschöpfungskette, eigene Darstellung

Eine gewisse „KI Literacy“, wie es im neuen DigComp 2.2 Rahmenwerk angelegt ist, sowie Computational Thinking als Problemlösungskompetenz (mit dem Computer denken) kann übergreifend für alle Entitäten eine zentrale Basis liefern. Damit könnten insbesondere auch die neuen Herausforderungen im Hinblick auf KI-ethische Entwicklung und Nutzung adressiert werden. Die Sensibilisierung für KI-Literacy und die Rechte der KI-Empfänger:innen werden folglich als wesentlich angesehen, um eine verantwortungsvolle KI-Einführung, insbesondere in der Zusammenarbeit mit einer generativen KI, zu fördern. Durch die Befähigung der Benutzer:innen, ihre Rechte zu verstehen und geltend zu machen, soll das Modell potenzielle ethische Herausforderungen mindern und faire und transparente KI-Praktiken gewährleisten. Zusammenfassend bietet die vorgeschlagene Struktur der KI-Wertschöpfungskette eine umfassende Perspektive auf die Rollen und Verantwortlichkeiten der verschiedenen Akteure im Prozess der KI-Entwicklung und -Nutzung. Dabei können „Prompting skills“ in der Zusammenarbeit mit generativer KI bereits als Allgemeinbildung sowie darüber hinaus als berufsübergreifende, transversale Kompetenzen verortet werden.

5 Fazit und Ausblick

Transversale Kompetenzen sind berufsübergreifend und werden sowohl im Arbeits- wie auch im Privatleben benötigt (vgl. Hart et al. 2021, 4ff.). Im DigComp Framework wird das Programmieren im Kompetenzbereich ‘digital content creation’ aufgeführt (Vuorikari et al. 2022, 8ff.). Die im DigComp definierten Kompetenzen sind sowohl für das Privatleben wie auch für den Arbeitstag relevant und können deshalb als Allgemeinbildung, respektive transversale Kompetenzen angesehen werden.

Die Art und Weise des Programmierens hat sich im Laufe der Zeit verändert. Berufslernende könnten in ihrem Arbeitsalltag mit drei verschiedenen Ansätzen des Programmierens in Berührung kommen. Der erste Ansatz ist das regelbasierte Programmieren. Das Coden mit einer traditionellen Programmiersprache ist nicht mehr zwingend, da Low-Code-Plattformen den Zugang zum regelbasierten Programmieren erleichtern. Mit dem zunehmenden Aufkommen von KI-Anwendungen kann der Begriff ‘Programmieren’ ausgedehnt werden. Berufslernende ‘programmieren’ die KI-Anwendung, indem sie ihre Fachexpertise in die Aufbietung von Trainingsdatensätzen einbringen und daher mit Informatikerinnen und Informatikern zusammen KI-Anwendungen mitentwickeln. Auch die vortrainierten Sprachmodelle, wie z. B. KI-Systeme auf der Basis von ChatGPT (bzw. GPT-3) erweitern das Verständnis des Programmierens. Solche Modelle können einerseits zum Erstellen von Computercode genutzt werden. Das heißt, sie erleichtern den Lernenden die Codeerstellung und können auch das Verständnis der Programmierung erhöhen. Andererseits werden die Berufslernenden aufgrund des Machine Learning-Paradigmas ‘in-Context Learnig’ zukünftig KI-Anwendungen selbständig durch ihre Nutzung weiterentwickeln können (vgl. Mc Cann et al. 2018; Brown et al. 2020). Alle drei Programmier-Ansätze fordern neue Kompetenzen von den Berufslernenden. Zentral für die Berufslernenden sind eine gut ausgebildete Daten-Affinität, ein Grundverständnis der KI-Technologien sowie neu auch ethische Überlegungen (z. B. Fairness, Datenverzerrungen, etc.) zu trainieren, um verantwortungsvoll bei der Entwicklung von KI-Systemen mitarbeiten zu

können. Mit dem Aufkommen generativer KI wie ChatGPT finden intensive Diskussionen statt, inwieweit Prompting Skills (verstanden als low coding skills) die Zusammenarbeit mit KI-Systemen prägen werden. In der Entwicklung von Services sind anspruchsvollere Kompetenzen im Sinne von Prompt Engineering aus der Fachexpertise erforderlich, in Zusammenarbeit mit Informatikern, welche das Finetuning großer Sprachmodelle übernehmen, um damit für eine Optimierung und Qualitätssteigerung zu sorgen.

Im 21. Jahrhundert entwickelt sich das Computational Thinking (CT) zu einer Kernkompetenz (vgl. Voogt et al. 2015). Methoden und Denkweisen der Informatik könne herangezogen werden, um Probleme in verschiedenen Domänen zu lösen. CT könnte also gemäß der Definition von ESCO als transversale Kompetenz bezeichnet werden. Verschiedene Forschende bringen CT in eine enge Beziehung mit der Kompetenz des Programmierens. Das Programmieren spricht mehrere Kernaspekte des CT an und beruht auf ähnlichen kognitiven Prozessen. (vgl. Lye/Koh 2014). Der Prozess der Codegenerierung mit Sprachmodellen wie ChatGPT ist dabei ähnlich wie bei der Verwendung für andere natürlichsprachliche Aufgaben. Daher bietet das Konzept CT u.E. großes Potenzial, als Universalsprache die Metakognition und Problemlösekompetenz von Lernenden zu erhöhen.

Als Synopse der Überlegungen dient eine KI-Wertschöpfungskette, um spezifischer auf die jeweiligen Programmierkompetenzen im Zeitalter der generativen KI einzugehen. Die Synopse dient als Entwurf und Diskussionsgrundlage. Für weiterführende Forschungsarbeiten ist zu klären, wie KI-ethische Aspekte stärker auf die jeweiligen Entitäten zu spezifizieren sind, um eine geteilte Verantwortungsübernahme zu klären (vgl. Hacker/Engel/Mauer 2023).

Zusammenfassend scheint uns die Frage, inwieweit weiterhin Programmierkenntnisse vermittelt werden sollten, da dies nun KI-Systeme effizienter übernehmen könnten, nicht gerechtfertigt. Aufgrund der vorgenommenen Analyse wird dafür argumentiert, dass vielmehr das Gegenteil der Fall ist. Berufslernende sind bereits heute an Prompting skills (im Sinne von low coding) heranzuführen, um diese auf künftige neue Zusammenarbeitsformen mit KI-Systemen vorzubereiten. Nach Ansicht der Autoren sollte der Ausgangspunkt der Frage vielmehr sein, welche „hybriden Schlüsselkompetenzen“ in neuen Mensch-Maschine Interaktionen künftig erforderlich sein werden.

Literatur

Angeli, C./Voogt, J./Fluck, A./Webb, M./Cox, M./Malyn-Smith, J./Zagami, J. (2016): A K-6 computational thinking curriculum framework: Implications for teacher knowledge. In: Journal of Educational Technology & Society, 19(3), 47-57.

Antonietti, C./Cattaneo, A./Amenduni, F. (2022): Can Teachers’ Digital Competence Influence Technology Acceptance in Vocational Education? In: Computers in Human Behavior, 132, Article 107266.

Battelle for Kids (2019): Framework for 21st century learning. Online: http://static.battelleforkids.org/documents/p21/P21_Framework_DefinitionsBFK.pdf (21.02.2023).

Bertelsmann Stiftung (2020): OECD-Lernkompass. Online: https://www.wissensatlas-bildung.de/publikation/oecd-lernkompass-2030/ (21.02.2023).

Bocconi, S./Chioccariello, A./Dettori, G./Ferrari, A./Engelhardt, K. (2016): Developing Computational Thinking in Compulsory Education-Implications for policy and practice. Luxembourg. Online: https://data.europa.eu/doi/10.2791/792158 (21.02.2023).

Bommasani, R./Hudson, D. A./Adeli, E./Altman, R./Arora, S./von Arx, S/.../Liang, P. (2021): On the Opportunities and Risks of Foundation Models. https://doi.org/10.48550/arXiv.2108.07258.

Brennan, K./Resnick, M. (2012): Using artifact-based interviews to study the development of computational thinking in interactive media design. Paper presented at annual American Educational Research Association meeting. Vancouver.

Brewer, L. (2013): Enhancing youth employability: What? Why? and How? Guide to core work skills. Geneva.

Brown, T./Mann, B./Ryder, N./Subbiah, M./Kaplan, J. D./Dhariwal, P./.../Amodei, D. (2020): Language models are few-shot learners. In: Advances in neural information processing systems, 33, 1877-1901.

Caliskan, A./Bryson, J./Narayanan, A. (2017): Semantics derived automatically from language corpora contain human-like biases. In: Science, 356, 183-186.

Christen, M./Mader, C./Čas, J./Abou-Chadi, T./Bernstein, A./Binder, N. B./.../Thouvenin, F. (2020): Wenn Algorithmen für uns entscheiden: Chancen und Risiken der künstlichen Intelligenz (Band 72). https://doi.org/10.3218/4002-9.

Delipetrev, B./Tsinaraki, C./ Kostić, U. (2020): AI watch, historical evolution of artificial intelligence: Analysis of the three main paradigm shifts in AI. EUR: Vol. 30221. Luxembourg. Online: https://publications.jrc.ec.europa.eu/repository/handle/JRC120469 (21.02.2023).

Di Ruscio, D./Kolovos, D./de Lara, J./Pierantonio, A./Tisi, M./Wimmer, M. (2022): Low-code development and model-driven engineering: Two sides of the same coin? In: Software and Systems Modeling, 21(2), 437-446.

Dilger, B. (2015): Kompetenzorientierung. Konsequenzen für die Unterrichtsgestaltung in curricularer, methodisch-didaktischer und prüfungsgestalterischer Hinsicht. In: Berufsbildung, 155, 2-5.

Europäische Gemeinschaften (2007): Schlüsselkompetenzen für lebenslanges Lernen – ein Europäischer Referenzrahmen. Luxemburg: Amt für amtliche Veröffentlichungen der Europäischen Gemeinschaften.

Finkbeiner, B./Schmitt, F. (2021): Künstliche Intelligenz in der Softwareentwicklung: Über die Schulter geschaut. In: iX Magazin für professionelle Informationstechnik, 8, 40-43.

Fraillon, J./Ainley, J./Schulz, W./Friedman, T./Duckworth, D. (2019): Preparing for Life in a Digital World: IEA International Computer and Information Literacy Study 2018 International Report. Amsterdam: International Association for the Evaluation of Educational Achievement

(IEA). Online: https://www.iea.nl/publications/study-reports/preparing-life-digital-world (21.02.2023).

Gimpel, H./Hall, K./Decker, S./Eymann, T./Lämmermann, L./Mädche, A./.../Vandrik, S. (2023): Unlocking the power of generative AI models and systems such asGPT-4 and ChatGPT for higher education. Hohenheim. Online: http://opus.uni-hohenheim.de/volltexte/2023/2146/ (14.06.2023).

Grover, S./Pea, R. (2017): Computational Thinking: A Competency Whose Time Has Come. Online: https://www.researchgate.net/publication/322104135_Computational_-Thinking_A_Competency_Whose_Time_Has_Come (21.02.2023).

Grover, S./Pea, R. (2013): Computational Thinking in K-12. In: A review of the state of the field. Educational Researcher, 42(1), 38-43.

Hacker, P./Engel, A./Mauer, M. (2023): Regulating ChatGPT and other Large Generative AI Models. In: FAccT '23: Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency, 1112-1123. https://doi.org/10.1145/3593013.3594067.

Handschuh S. (2023): Potentiale und Schwächen von ChatGPT. Online: https://www.youtube.com/watch?v=_UB1vD4ZhTY (21.02.2023).

Hart, J./Noack, M./Plaimauer, C./Bjørnåvold, J. (2021): Towards a structured and consistent terminology on transversal skills and competences. Brüssel.

Harteis, C./Fischer, C. (2020): Wissensmanagement unter Bedingungen von Arbeit 4.0. In: Maier, G. W./Engels, G./Steffen, E. (Hrsg.): Handbuch Gestaltung digitaler und vernetzter Arbeitswelten. Wiesbaden, 267-284.

Herft, A. (2023): A Teacher's Prompt Guide to ChatGPT aligned with 'What Works Best' Guide. Online: https://drive.google.com/file/d/15qAxnUzOwAPwHzoaKBJd8FAgiOZYcIxq/view (14.06.2023).

Hsu, T. C./Chang, S. C./Hung, Y. T. (2018): How to learn and how to teach computational thinking: Suggestions based on a review of the literature. In: Computers & Education, 126, 296-310.

Ifenthaler, D./Gibson, D./Prasse, D./Shimada, A./Yamada, M. (2021): Putting learning back into learning analytics: actions for policy makers, researchers, and practitioners. In: Educational Technology Research and Development, 69, 2131-2150. https://doi.org/10.1007/s11423-020-09909-8.

Israel, M./Pearson, J. N./Tapia, T./Wherfel, Q. M./Reese, G. (2015): Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. In: Computers & Education, 82, 263-279.

Kirschner, P. A./van Merriënboer, J. J.G. (2013): Do Learners Really Know Best? Urban Legends in Education. In: Educational Psychologist, 48(3), 169-183. https://doi.org/10.1080/00461520.2013.804395.

Lamprou, A./Repenning, A. (2018): Teaching How to Teach Computational Thinking. the 23rd Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE'18), Larnaca. Online: https://wiki.computationalthinkingfoundation.org/wiki/images/9/92/Iticse18main-id125-p.pdf (14.06.2023).

LeCun, Y./Bengio, Y./Hinton, G. (2015): Deep learning. In: Nature, 521(7553), 436-444. https://doi.org/10.1038/nature14539.

Lye, S. Y./Koh, J. H. L. (2014): Review on teaching and learning of computational thinking through programming: What is next for K-12? In: Computers in Human Behavior, 41, 51-61.

Makridakis, S. (2017): The forthcoming Artificial Intelligence (AI) revolution: Its impact on society and firms. In: Futures, 90, 46-60.

McCann, B./Keskar, N. S./Xiong, C./Socher, R. (2018): The natural language decathlon: Multitask learning as question answering. arXiv preprint arXiv:1806.08730.

Mertens, D. (1974): Schlüsselqualifikationen. Thesen zur Schulung für eine moderne Gesellschaft. In: Mitteilungen aus der Arbeitsmarkt und Berufsforschung, 7(1), 36-43.

Microsoft (2023): Was ist Microsoft Power Platform? Online: https://docs.microsoft.com/dede/learn/modules/introduction-power-platform/2-what-is-power-platform (12.02.2023).

OECD (2005): Definition und Auswahl von Schlüsselkompetenzen. Zusammenfassung. Paris.

OpenAI (2023): Generative Models. Online: https://openai.com/blog/generative-models/ (21.02.2023).

Pollak, M/Ebner, M. (2019): The Missing Link to Computational Thinking. In: Future Internet 11(12), 263-278. https://doi.org/10.3390/fi11120263.

Potts, C. (2023): Stanford Webinar - GPT-3 & Beyond. Online: https://www.youtube.com/watch?v=-lnHHWRCDGk (21.02.2023).

Ratto, M. (2011): Critical making: Conceptual and material studies. In: The Information Society: An International Journal, 27, 252-260.

Rauseo, M./Harder, A./Glassey-Previdoli, D./Cattaneo, A./Schumann, S./Imboden, S. (2022): Same, but Different? Digital Transformation in Swiss Vocational Schools from the Perspectives of School Management and Teachers. In: Technology, Knowledge and Learning, 28, 407-427. https://doi.org/10.1007/s10758-022-09631-9.

Repenning, A. (2018): Design-based Research und Computational Thinking (Unveröffentlichtes Papier zur Forschungsstrategie). Windisch.

Repenning, A. (2019): Warum Interdisziplinarität die Grundlage für Computational Thinking ist? Online: https://www.societybyte.swiss/2019/07/08/computional-thinking/ (21.02.2023).

Rode, J. A./Weibert, A./Marshall, A./Aal, K./von Rekowski, T./Elmimouni, H./Booker, J. (2015): From Computational Thinking to Computational Making. In: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing. Osaka, 239-250.

Royal Society (2012): Shut down or restart: The way forward for computing in UK schools. Online: http://royalsociety.org/education/policy/computing-in-schools/report (21.02.2023).

Scharnhorst, U. (2021): Transversale Kompetenzen–notwendig, erwünscht und schwierig zu erreichen. In: Berufsbildung in Wissenschaft und Praxis, 50(1), 18-23.

Scharnhorst, U./Kaiser, H. (2018): Transversale Kompetenzen für eine ungewisse digitale Zukunft. Digitalisierung und Berufsbildung. Herausforderungen und Wege in die Zukunft. OBS EHB Trendbericht, 3.

Scharnhorst, U./Kaiser, H. (2018): Transversale Kompetenzen: Bericht im Auftrag des Staatssekretariats für Bildung, Forschung und Innovation SBFI im Rahmen des Projekts" Berufsbildung 2030-Vision und Strategische Leitlinien". Bern.

Scherer, R./Siddiq, F./Sánchez Viveros, B. (2019): The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. In: Journal of Educational Psychology, 111(5), 764.

Seufert, S. (2017): Digital competences Paper commissioned by the Swiss Science and Innovation Council SSIC. Online: https://www.swir.ch/images/stories/pdf/en/Exploratory_study_3_2017_Excerpt_Digital_Competences_SSIC_EN.pdf (21.02.2023).

Shute, V. J./Sun, C./Asbell-Clarke, J. (2017): Demystifying computational thinking. In: Educational research review, 22, 142-158.

Stanford University (2016): Artificial intelligence and life in 2030. One hundred year study on artificial intelligence: report of the 2015-2016 study panel. Stanford CA. Online: https://ai100.stanford.edu/sites/g/files/sbiybj9861/f/ai_100_report_0831fnl.pdf (21.02.2023).

Voogt, J./Fisser, P./Good, J./Mishra, P./Yadav, A. (2015): Computational thinking in compulsory education: Towards an agenda for research and practice. In: Education and Information Technologies, 20, 715-728. Online: https://link.springer.com/article/10.1007/s10639-015-9412-6 (21.02.2023).

Vuorikari, R./Kluzer, S./Punie, Y. (2022): DigComp 2.2: The Digital Competence Framework for Citizens – With new examples of knowledge, skills and attitudes (No. JRC128415). Joint Research Centre (Seville site).

Vuorikari, R./Holmes, W. (2022): DigComp 2.2. Annex 2. Citizens interacting with AI systems. Luxembourg.

Waszkowski, R. (2019): Low-code platform for automating business processes in manufacturing. In: IFAC-PapersOnLine, 52(10), 376-381.

Weinert, F. E. (2001): Vergleichende Leistungsmessung in Schulen – eine umstrittene Selbstverständlichkeit. In: Weinert, F. E. (Hrsg.): Leistungsmessungen in Schulen. Weinheim/Basel, 17-31.

Wey-Han, T. (2011). Seymour Papert und Konstruktionismus. Online: http://blogs.epb.uni-hamburg.de/spielendlernen2011/?p=562 (21.02.2023)

Wing, J. M. (2006): Computational thinking. In: Communications of the ACM, 49(3), 33-35. https://doi.org/10.1145/1118178.1118215.

Yadav, A./Ocak, C./Oliver, A. (2022): Computational thinking and metacognition. In: TechTrends, 66(3), 405-411.

Yadav, A./Good, J./Voogt, J./Fisser, P. (2018): Computational Thinking as an Emerging Competence Domain. In: M. Mulder (ed.): Technical and Vocational Education and Training: Vol. 23. Competence-based Vocational and Professional Education: Bridging the Worlds of Work and Education. Cham, 1051-1067.

Yang, Q./Suh, J./Chen, N. C./Ramos, G. (2018): Grounding Interactive Machine Learning Tool Design in How Non-Experts Actually Build Models. In: DIS '18: Proceedings of the 2018 Designing Interactive Systems Conference, 573-584.

Zitieren dieses Beitrags

Seufert, S./Spirgi, L. (2023): Programmieren im Zeitalter der generativen KI: eine transversale Kompetenz in der Berufsbildung? In: *bwp@* Spezial 20: Die Förderung von transversalen Kompetenzen in der Berufsbildung, hrsg. v. Barabasch, A./Fischer, S., 1-26. Online: https://www.bwpat.de/spezial20/seufert_spirgi_spezial20.pdf (19.11.2023).

Die Autor*innen

Prof. Dr. SABINE SEUFERT

Universität St. Gallen, Institut für Bildungsmanagement und Bildungstechnologien

St.Jakob-Strasse 21, CH-9000 St.Gallen

sabine.seufert@unisg.ch

www.ibb.unisg.ch

LUKAS SPIRGI

Universität St. Gallen, Institut für Bildungsmanagement und Bildungstechnologien

St.Jakob-Strasse 21, CH-9000 St.Gallen

lukas.spirgi@unisg.ch

www.ibb.unisg.ch