L. Hellmig, M. Hennecke (Hrsg.): INFOS23,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2023 11

Softwarequalität im Informatikunterricht? Eine Auswertung nationaler und internationaler Curricula

Elena Starke[1], Tilman Michaeli[2]

Abstract: Mangelnde Softwarequalität erleben Schüler:innen regelmäßig in ihrem Alltag, sei es in Form von Fehlermeldungen in Anwendungssoftware, Systemausfällen oder Medienberichten zu Sicherheitslücken. Es besteht jedoch kein Konsens darüber, inwieweit Softwarequalität Teil des Informatikunterrichts sein sollte. Einerseits können durch Unterricht zum Thema Softwarequalität solche Alltagsphänomene erklärbar gemacht und darüber hinaus spezielle informatische Denkweisen vermittelt werden. Andererseits wird Softwarequalität dem ingenieurwissenschaftlichen Teil der Informatik zugerechnet, der als nur stark eingeschränkt allgemeinbildend angesehen wird und es stellt sich die Frage, wie die entsprechenden Inhalte den Schüler:innen auf eine handlungsorientierte und motivierende Art zugänglich gemacht werden können. In diesem Beitrag wird daher anhand einer Analyse aktueller nationaler und internationaler Bildungspläne der Stellenwert von Softwarequalität für den Informatikunterricht diskutiert. Dazu werden Curricula mittels qualitativer Inhaltsanalyse mit einer fachwissenschaftlichen Charakterisierung des Themengebietes abgeglichen. Die Ergebnisse zeigen, dass ein Schwerpunkt auf Codequalität und der Vermittlung von Kompetenzen wie dem Testen, Analysieren und Bewerten von Software liegt. Im Gegensatz dazu finden Aspekte wie das Messen von oder die Definition von Anforderungen an die Softwarequalität nur wenig Beachtung.

Keywords: Softwarequalität; Curricula; Inhaltsanalyse; Allgemeinbildung; Schulunterricht

1 Einleitung

Softwareprodukte sind im täglichen Leben der Schülerinnen und Schüler allgegenwärtig. Jedoch gehören auch Probleme wie Bluescreens, Fehlermeldungen oder das Einfrieren der Anwendung zum Umgang mit Software dazu. Diese Phänomene sind dem Bereich Softwarequalität zuzuordnen, einem bisher weitgehend unbearbeiteten Feld informatischer Bildung in Schulen.

Zunächst ist jedoch zu diskutieren, ob und wenn ja, welche Aspekte von Softwarequalität im Rahmen von Informatikunterricht thematisiert werden sollen: Einerseits bietet die eigenständige Entwicklung von kleineren Programmen die Möglichkeit auf praktische Art und Weise zu veranschaulichen, dass Fehler in der Softwareentwicklung unvermeidbar sind. Im Zuge dessen ermöglicht Informatikunterricht die Beantwortung der Frage, warum die Entwicklung von Software, die zuverlässig und fehlerfrei arbeitet, so komplex ist. Darüber

[1] TU München, Didaktik der Informatik, Arcisstr. 21, 80333 München, elena.starke@tum.de
[2] TU München, Didaktik der Informatik, Arcisstr. 21, 80333 München, tilman.michaeli@tum.de

CC BY NC SA

hinaus bietet sich die Gelegenheit, sich mit ausgewählten weiteren Herausforderungen bei der Entwicklung technischer Systeme auseinanderzusetzen [19]. So kann ein Beitrag zur Welterklärung geleistet werden, der selbst aus medienpädagogischer Perspektive hervorgehoben wird [18]. Des Weiteren erhalten die Schülerinnen und Schüler durch das gezielte Reflektieren und Bewerten von Lösungen die Möglichkeit, alternative Herangehensweisen an Probleme kennen zu lernen und so ihre Problemlösefähigkeiten zu erweitern. Diese Kompetenzen können beim Programmieren lernen unterstützen und im Anschluss auf eigene Projekte angewendet werden [10].

Andererseits wird Softwareentwicklung und somit Softwarequalität dem ingenieurwissenschaftlichen Teil der Informatik zugeordnet, der als nur stark eingeschränkt allgemeinbildend angesehen wird [3]. Angesichts zu kleiner Projekte im Informatikunterricht und oft zu kurzer Projektphasen stellt sich weiterhin die Frage, wie das Thema Softwarequalität so aufbereitet werden kann, dass die Schülerinnen und Schüler das erworbene Wissen anwenden und die Relevanz der Inhalte selbst erfahren können - eine Herausforderung selbst in der universitären Ausbildung [7].

Bildungspläne können Aufschluss darüber geben, welche Aspekte von Softwarequalität im Informatikunterricht aktuell berücksichtigt werden. Ziel dieses Beitrags ist daher die Analyse ausgewählter nationaler wie internationaler Curricula und der Vergleich mit einer fachwissenschaftlichen Charakterisierung als Grundlage für die Diskussion des Stellenwerts von Softwarequalität im Informatikunterricht.

2 Hintergrund und Forschungsstand

Softwarequalität beschreibt eine Vielzahl von anwendungsspezifischen Qualitätseigenschaften, wie beispielsweise Funktionalität, Effizienz oder Benutzerfreundlichkeit. Anhand dieser Merkmale kann die Qualität von Softwareprodukten charakterisiert werden. Zur Sicherung der Softwarequalität bietet sich den Einsatz verschiedener Techniken wie zum Beispiel Testen, Debuggen oder Analysieren einer Lösung an. Darüber hinaus können auch Werkzeuge den Qualitätssicherungsprozess unterstützen. Hierfür empfiehlt sich der Einsatz von geeigneten Entwicklungs- und Testumgebungen oder eines Systems zur Versionsverwaltung [12].

Betrachtet man den Stand der Forschung, ist festzustellen, dass es kaum Arbeiten gibt, die sich mit dem Thema Softwarequalität im Bereich der Schulinformatik beschäftigen. Bislang wurde lediglich Codequalität – das Maß, in dem ein Code vorgegebene Anforderungen erfüllt – als Teilaspekt der Softwarequalität untersucht. So hat beispielsweise die Evaluation eines MOOCs für Kinder, der neben ersten Programmierkenntnissen auch grundlegende Aspekte der Codequalität vermitteln sollte, gezeigt, dass Techniken zur Verbesserung der Codequalität nicht schwieriger zu verstehen sind als allgemeine Programmierprinzipien [11]. Des Weiteren zeigte ein Experiment mit 12-14-jährigen Programmieranfängerinnen und -anfänger, dass die Fähigkeit, die Funktionsweise eines Programms zu erklären, durch Code

Smells – Indizien, die auf Verbesserungspotenzial im Code hinweisen – nicht beeinflusst wird. Dagegen wird die Modifikation von Programmen durch enthaltene Code Smells nachweislich erschwert [10]. Bezüglich der Vermittlung von Codequalität im Schulunterricht wurde festgestellt, dass Lehrkräfte nicht zwischen Codequalität und Code Style, also Konventionen, wie der Code gestaltet werden sollte, unterscheiden [16] aber viele gute und problematische Praktiken im Zusammenhang mit Code Style beschreiben können, jedoch nicht unbedingt in der Lage sind, diese in einem gegebenen Codebeispiel wiederzuerkennen [14].

Vorwiegend wird Softwarequalität in hochschuldidaktischer Forschung adressiert. Auch hier stellt Codequalität einen Fokus der Forschung dar: So konnte beispielsweise ein Zusammenhang zwischen der Kreativität der Lösung und mangelnder Codequalität festgestellt [9] oder verschiedene semantische Stilprobleme identifiziert werden, die auf fehlende Kenntnisse bezüglich eines bestimmten Aspekts der Programmierung hinweisen können [4]. Ferner wurde ein Gamification-Ansatz erfolgreich pilotiert, um die Motivation der Studierenden zu steigern, qualitativ besseren Code zu schreiben [13], obwohl viele Programmierkurse für Anfänger Codequalität nicht in ihren Lernzielen erwähnen [15]. Darüber hinaus werden aber auch andere Aspekte der Softwarequalität untersucht: Insbesondere zur Vermittlung von Testfähigkeiten wurden zahlreiche Untersuchungen durchgeführt. Die zeitliche Belastung der Dozierenden und die häufig geringe Motivation der Studierenden stellt eine Herausforderung bei der Vermittlung von Testkompetenzen dar. Um dieser zu begegnen, wurden unterschiedliche Lehransätze evaluiert, wie z.B. die Integration von Testen in einen regulären Programmierkurs oder das Angebot eines Spezialisierungskurses sowie verschiedene spezifische Tools, die die Vermittlung von Testkompetenzen unterstützen können, wie z.B. WebCAT und CodeDefenders [7]. Neben der Integration von Code Review in Softwareenentwicklungskurse mit Hilfe von Code Review Notebooks [5], wurden Konzepte zur Vermittlung von Code Review auf Basis von Gamification entwickelt [1].

3 Methodik

Ziel dieses Beitrags ist es, einen ersten Überblick über die Repräsentation von Softwarequalität im Informatikunterricht zu erhalten und damit die Grundlage für weitere Arbeiten in diesem Bereich zu legen. Dazu wurden nationale Curricula der Bundesländer, Bildungsstandards der Gesellschaft für Informatik (GI) für die Sekundarstufen I und II, einheitliche Prüfungsanforderungen in der Abiturprüfung Informatik (EPA) und englischsprachige Curricula aus Neuseeland (NZC), den Vereinigten Staaten (USA) und Israel (IS) [6] analysiert. Die Dokumente wurde mit Hilfe einer strukturierten qualitativen Inhaltsanalyse nach [17] untersucht. Das Kategoriensystem wurde deduktiv auf Basis der Charakterisierung von Softwarequalität aus dem *Guide to the Software Engineering Body of Knowledge* (SWEBOK) [2] gewählt. Das deduktive Vorgehen bietet die Möglichkeit Unterschiede zwischen fach- und bildungswissenschaftlich relevanten Inhalten zu identifizieren, während eine rein induktive Entwicklung des Kategoriensystems lediglich den Status quo in den Schulen widerspiegeln würde. Durch induktive Ergänzungen kann sichergestellt werden, dass alle in den Curricula

enthaltenen Aspekte in das Kategoriensystem aufgenommen werden können [8]. Dafür wurde für jeden Code, der keiner bestehenden Kategorie zugeordnet werden konnte, eine neue hinzugefügt, welche abschließend in das bisherige System eingeordnet wurde.

Die im SWEBOK beschriebenen übergeordneten Bereiche von Softwarequalität sind:

- **Grundlagen der Softwarequalität:**
  Grundlegende Eigenschaften, die für die Beschäftigung mit Softwarequalität relevant sind
- **Softwarequalitätsmanagement-Prozess:**
  Definition von Prozessen, die sicherstellen können, dass die Produkte und Entwicklungsprozesse den geforderten Qualitätsanforderungen entsprechen
- **Praktische Überlegungen:**
  Aktive Tätigkeiten, die dazu beitragen, dass die Softwarequalität den geforderten Anforderungen entspricht
- **Softwarequalitätswerkzeuge:**
  Technische Hilfsmittel zur Einhaltung der geforderten Softwarequalität

Aufgrund des unterschiedlichen Detailgrades in der Ausgestaltung der Lehrpläne wird das Auftreten einer Kodierung pro Dokument als Wahrheitswert kodiert. Deswegen ist die Länge der kodierten Textstellen ist nicht beschränkt und hat auch keinen Einfluss auf das Analyseergebnis. Abbildung 1 gibt einen Überblick über das finale Kategoriensystem. Die induktive Erweiterung der Kategorien wurden mit anderen Mitgliedern der Arbeitsgruppe diskutiert und evaluiert.

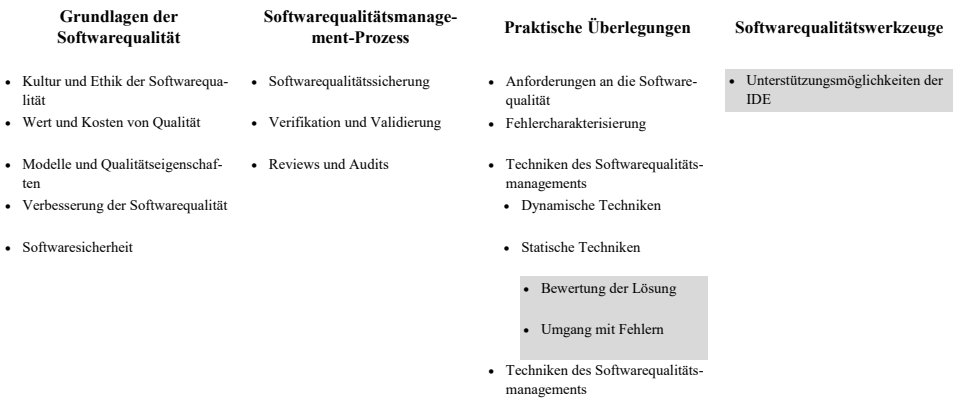

Abb. 1: Finales Kategoriensystem: induktiv ergänzte Kategorien sind grau hinterlegt

4 Ergebnisse

Im Folgenden werden die Ergebnisse der Auswertung der Curricula dargestellt. Dabei werden zunächst die verschiedenen Kategorien anhand konkreter Beispiele aus dem Korpus beleuchtet, bevor Unterschiede bezüglich der Verteilung auf die Sekundarstufe 1 und 2 zusammengefasst werden. Eine Übersicht über die Ergebnisse der Analyse bietet Abbildung 2.

	EPA	GI	BW	BY	BE & BB	HB	HE	HH	MV	NI	NW	RP	SL	SN	ST	SH	TH	NZC	IS	USA
Grundlagen der Softwarequalität																				
• Kultur und Ethik der Softwarequalität			1									2								
• Wert und Kosten von Qualität			1																	
• Modelle und Qualitätseigenschaften		2	1	2	2				1/2	1	1/2	1/2		2	2	1/2		2	1/2	1/2
• Verbesserung der Softwarequalität	2	1/2		2			1		1/2		2	1		1	1/2	1	1/2	1/2	1	1
• Softwaresicherheit			2									2								1/2
Softwarequalitätsmanagement-Prozess																				
• Softwarequalitätssicherung				2											2			2		1
• Verifikation und Validierung			1/2						1/2	1	1/2	2			2	1			1	
• Reviews und Audits		1/2	1/2	2	1				1	2	1/2				1			2		1
Praktische Überlegungen																				
• Anforderungen an die Softwarequalität		1/2	1									2						1/2	2	1
• Fehlercharakterisierung			1/2		1		1		1		2	1/2				1		1/2	1	
• Techniken des Softwarequalitätsmanagements																				
Dynamische Techniken	2	1/2	1/2	1/2	1/2	2	1/2	2	1/2	1	1/2	1/2		1	1/2	1/2	2	1/2	1/2	1/2
Statische Techniken																				
Bewertung der Lösung		1/2	1/2	2	2	2	1	2	2	1/2	1/2	1/2			1/2	2				
Umgang mit Fehlern		1	1/2				1		1		2				1/2	1	1/2	2	1	1
• Messung der Softwarequalität																				
Softwarequalitätswerkzeuge																				
• Unterstützungsmöglichkeiten der IDE		1	1/2		2		1		1		2	2			1/2	1/2	1/2	2	2	

Abb. 2: Überblick über die Ergebnisse der Kodierung: Die Verteilung der Inhalte auf Sekundarstufe 1 und 2 ist durch die Ziffern in den entsprechenden Zellen angegeben

Grundlagen der Softwarequalität In den analysierten Curricula werden Aspekte der *Kultur und Ethik der Softwarequalität* nur selten aufgegriffen. Ein Beispiel aus dem Korpus

ist die Diskussion um die Frage, wer für die Folgen von Softwarefehlern im medizinischen Bereich haftet. Ebenso werden *Wert und Kosten von Qualität* nur selten kodiert. Auf diese Aspekte wird lediglich im Zusammenhang mit der Diskussion um den Kompromiss zwischen Effizienz und Robustheit von Fehlerkorrekturverfahren eingegangen. Weiterhin steht auch der Bereich der *Softwaresicherheit* nicht im Fokus. Gelegentlich können Punkte im Korpus ausgemacht werden, die darauf hindeuten, dass verschiedene Sicherheitsmaßnahmen unter Berücksichtigung anderer Qualitätsmerkmale verglichen werden sollen. Im Gegensatz zu den vorherigen Aspekten werden in mehreren Curricula durchaus Kompetenzerwartungen identifiziert, die sich mit *Modellen und Qualitätseigenschaften* von Software befassen. Allerdings stehen Qualitätseigenschaften wie Funktionalität, Korrektheit und Effizienz im Mittelpunkt, während weitere interne und externe Merkmale, wie beispielsweise Zuverlässigkeit, Benutzbarkeit und Übertragbarkeit und die Anforderung, dass „bei der Entwicklung eigener Softwareprodukte exemplarisch Qualitätsmerkmale“ beachtet werden sollen (RP) nur vereinzelt kodiert werden konnten. Qualitätsmodelle – im Sinne internationaler Standards – können in den analysierten Dokumenten nicht gefunden werden. Darüber hinaus wurden Aspekte, die die *Verbesserung der Softwarequalität* betreffen, kodiert. Der Schwerpunkt dabei liegt auf dem Erkennen und Beheben von Fehlern, dem Testergebnisse bewerten und der Berücksichtigung von Feedback der Teammitglieder.

Softwarequalitätsmanagement-Prozess Im Bereich der *Softwarequalitätssicherung* weisen die Curricula nur wenige Aspekte auf, die Kompetenzen in diesem Bereich vermitteln sollen. Ein Beispiel, das kodiert wurde, ist die Dokumentation von Programmen zur späteren Nachvollziehbarkeit. Darüber hinaus ist die Auseinandersetzung mit den Prozessen *Verifikation und Validierung* und *Reviews und Audits* nur in wenige Curricula integriert. In diesen Kategorien wurden Beispiele wie das Verifizieren eines Ergebnisses, das schrittweise nachvollziehen von Algorithmen, das Analysieren von Programmen oder das Finden von Fehlern durch bloße Betrachtung des Codes kodiert.

Praktische Überlegungen Nur in wenigen Fällen wurden Lehrplanelemente kodiert, die dem Bereich *Anforderungen an die Softwarequalität* zugeordnet werden können: Als zu erwerbende Kompetenzen können im Korpus einerseits auf der Ebene des Codes Kommentare im Quellcode, eine logische Programmstruktur und ein klarer, präziser Code gefunden werden. Andererseits sollen Aspekte wie Funktionalität, Kosten und Geschwindigkeit bei der Erstellung von Softwareprodukten berücksichtigt werden. Des Weiteren werden in etwa der Hälfte der analysierten Dokumente im Rahmen der *Fehlercharakterisierung* zum einen typische Fehler (z.B. Objekt wird verwendet, aber nicht erzeugt; Unterschied zwischen Wertzuweisung und Vergleich) und bekannte Bugs vor allem auf Codeebene diskutiert. Zum anderen wird der „Unterschied zwischen verschiedenen Fehlerarten (Comilerfehler/Laufzeitfehler und syntaktisch/semantisch)“ erläutert (BW). Die *Techniken des Softwarequalitätsmanagements* sind die prominentesten Aspekte der Analyse. Diese gliedern sich in dynamische und statische Techniken. Zu den *dynamischen Techniken* gehören

alle Aktivitäten, die eine Ausführung des Codes verlangen, also insbesondere Testen. Ein Beispiel hier ist das „Überprüfen der Funktionalität eines vorgegebenen Softwareprodukts mit unterschiedlichen Testverfahren“ (BY). Die *statischen Techniken*, d.h. die Verfahren, die ohne Ausführung des Codes auskommen, sind in zwei Unterkategorien unterteilt und ebenfalls sehr häufig in den Lehrplänen vertreten. Das *Bewerten der Lösung* beschreibt hier all die Kompetenzen, die die Schülerinnen und Schüler erwerben sollen, um „die Angemessenheit von Lösungen und die erreichten Resultate“ (BW) und „die Eignung und Qualität des entwickelten Produkts“ zu bewerten (HE). Außerdem sollen sie lernen zu „überprüfen, ob eine Implementierung die Problemstellung löst“ (NI). Dabei ist festzuhalten, dass sich die Kategorie *Bewerten der Lösung* auf die Qualitätsbewertung der Gesamtlösung bezieht, daher wurden etwa Kompetenzerwartungen, die sich lediglich auf die Effizienzbewertung einzelner Algorithmen beziehen, nicht kodiert – auch wenn Effizienz ein relevantes Qualitätsmerkmal der Softwarequalität ist. Die zweite Unterkategorie stellt den *Umgang mit Fehlern* dar. In dieser Kategorie werden Kompetenzen aus dem praktischen Fehlermanagement kodiert. Beispiele sind vor allem „Fehler gegebenenfalls systematisch suchen“ (RP), „eine Fehleranalyse durchführen“ (TH) und die „Fehlerbehandlung“ (SH). Des Weiteren sollen die Schülerinnen und Schüler „Fehlermeldungen der Entwicklungsumgebung [...] nutzen, um Programme fehlerfrei zu implementieren“ (BW). Im Gegensatz dazu ist die *Messung der Softwarequalität* in den Curricula unserer Stichprobe nicht vorgesehen.

Softwarequalitätswerkzeuge Insbesondere didaktische Entwicklungsumgebungen und deren Funktionen und Möglichkeiten stellen technische Hilfsmittel dar, die der Kategorie *Unterstützungsmöglichkeiten der IDE* zugeordnet werden können. Durch das Bereitstellen von Testumgebungen und Debuggern können Entwicklungsumgebungen dazu beitragen, dass die Schülerinnen und Schüler Strategien entwickeln, „um fehlerfreien Code [zu] schreiben“ (BW) und somit die Funktionalität ihres Programms zu verbessern.

Hinsichtlich der Verteilung der identifizierten Lehrplaninhalte auf die Jahrgangsstufen lässt sich festhalten, dass Elemente der Softwarequalität in allen Jahrgangsstufen, in denen Informatik unterrichtet wird, zu finden sind. Allerdings ist in der Sekundarstufe 1 in den analysierten Curricula ein Fokus auf das Auffinden von Fehlern durch altersgemäße Testmöglichkeiten und damit die praktische Ermittlung der Korrektheit festzustellen. Weiterhin werden im Verlauf der Sekundarstufe 1 verstärkt (didaktische) Entwicklungsumgebungen genannt, deren Möglichkeiten gewinnbringend eingesetzt werden sollen. Außerdem werden sowohl Kompetenzerwartungen für die Diskussion von Fehlern und deren Folgen identifiziert als auch die Ergebnisse systematischer Tests zur Ableitung von Änderungen und zur Bewertung der Lösung herangezogen. Die theoretische Betrachtung, insbesondere die Diskussion von Qualitätsmerkmalen, findet hingegen vor allem in der Sekundarstufe 2 statt.

5 Diskussion und Fazit

Anhand einer qualitativen Analyse wurde untersucht, welche Aspekte von Softwarequalität in deutschen und internationalen Curricula enthalten sind. Die Ergebnisse zeigen, dass Bereiche der Softwarequalität in fast ausnahmslos allen Jahrgangsstufen und Curricula der untersuchten Stichprobe thematisiert werden. Allerdings ist ein starker Fokus auf Aspekte der Codequalität festzustellen. Weiterhin weisen die Curricula einen Schwerpunkt bezüglich bestimmter *Techniken des Softwarequalitätsmanagements* wie Testen, Bewerten der Lösung und den Umgang mit Fehlern auf.

Im Vergleich zur fachwissenschaftlichen Strukturierung des Themenfeldes, die als Kategoriensystem genutzt wurde, konnte eine deutliche Überschneidung der Inhalte festgestellt werden, jedoch sind die Kategorien keineswegs disjunkt. Insbesondere finden sich Elemente der *Verbesserung der Softwarequalität* auch in verschiedenen Unterkategorien der Kategorie *Praktische Überlegungen*. Des Weiteren kann festgestellt werden, dass die *Bewertung der Lösung* in vielen Curricula ein Bestandteil ist, jedoch ohne dass zuvor explizit *Anforderungen an die Softwarequalität* definiert wurden.

Bezüglich der Verortung der Inhalte zeigen die Ergebnisse, dass bereits in der Sekundarstufe 1 Aspekte der Softwarequalität vermittelt werden sollen. Die Curricula für diese Altersstufe legen den Schwerpunkt auf praktische Aspekte, wie z.B. die ausführliche Thematisierung von Fehlern und deren Erkennung, Analyse und Behebung. Die Tatsache, dass diese Inhalte in allen Jahrgangsstufen kodiert wurden, lässt darauf schließen, dass die Schülerinnen und Schüler einen souveränen Umgang mit diesen Themen erlernen sollen und diese als relevant für die Allgemeinbildung angesehen werden. Aspekte wie die theoretische Betrachtung von Qualitätseigenschaften finden sich hingegen eher in den in den Curricula der Sekundarstufe 2 und werden damit als nur eingeschränkt allgemeinbildend eingestuft.

Bei der Betrachtung des internationalen Vergleichs sind nur zwei Unterschiede nennenswert: Zum einen ist die in deutschen Curricula sehr prominente *Bewertung der Lösung* in unserer internationalen Stichprobe nicht vorhanden. Zum anderen wird dem Aspekt der Definition von *Anforderungen an die Softwarequalität* in der internationalen Stichprobe mehr Aufmerksamkeit gewidmet als in den nationalen Lehrplänen.

Im Vergleich mit Forschungsschwerpunkten im Bereich Hochschuldidaktik zeigt sich, dass Ähnlichkeiten bezüglich der Vermittlung *dynamischer Techniken* wie z.B. Testen im bestehen. Ebenso findet man die Förderung von Fähigkeiten wie Code Review in den untersuchten Curricula. Dort wird es im Rahmen von *Reviews und Audits* und *Bewertung von Lösungen* vermittelt.

Ziel dieser Arbeit war es, die aktuelle Repräsentation von Softwarequalität im Informatikunterricht in nationalen wie internationalen Curricula zu untersuchen. Die Analyse zeigt, dass Aspekte der Softwarequalität an vielen Stellen in den Lehrplänen vertreten sind, jedoch mit einem starken Fokus auf Codequalität. Außerdem gibt sie Aufschluss darüber, dass vor allem die Bereiche der Softwarequalität, die sich auf die Codequalität beziehen,

durchaus als allgemeinbildend angesehen werden können, während theoretische Aspekte der Softwarequalität als eher eingeschränkt für die Allgemeinbildung relevant eingestuft werden können.

Wie die entsprechenden Inhalte aber nun konkret im Unterricht umgesetzt werden, konnte im Rahmen dieser Analyse nicht ermittelt werden und muss in weiterführenden Arbeiten betrachtet werden. Insbesondere stellt die Vermittlung dieser eine große Herausforderung dar, sodass in Zukunft handlungsorientierte und motivierende Herangehensweisen an die Thematik Softwarequalität identifiziert werden sollen.

Literatur

[1] Baris Ardic, Irem Yurdakul und Eray Tuzun. „Creation of a Serious Game for Teaching Code Review: An Experience Report“. In: *2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T)*. 2020.

[2] Pierre Bourque und Richard E. Fairley. *Guide to the software engineering body of knowledge version 3.0 Swebok®: A project of the IEEE Computer Society*. Piscataway, 2014.

[3] Jürgen Burkert. „Informatikunterricht — Quo vadis? Thesen zu Stand und Entwicklung der Schul-Informatik“. In: *Innovative Konzepte für die Ausbildung*. Hrsg. von W. Brauer und Sigrid Schubert. Informatik aktuell. 1995.

[4] Giuseppe De Ruvo u. a. „Understanding semantic style by analysing student code“. In: *Proceedings of the 20th Australasian Computing Education Conference*. Hrsg. von Raina Mason und Simon. 2018.

[5] Juan Carlos Farah u. a. „Toward Code Review Notebooks“. In: *2022 International Conference on Advanced Learning Technologies (ICALT)*. 2022.

[6] Judith Gal-Ezer und David Harel. „Curriculum and Course Syllabi for a High-School CS Program“. In: *Computer Science Education* 9.2 (1999).

[7] Vahid Garousi u. a. „Software-testing education: A systematic literature mapping“. In: *Journal of Systems and Software* 165 (2020).

[8] Andreas Grillenberger und Ralf Romeike. „A comparison of the field data management and its representation in secondary CS curricula“. In: *Proceedings of the 9th Workshop in Primary and Secondary Computing Education on - WiPSCE '14*. Hrsg. von Carsten Schulte, Michael E. Caspersen und Judith Gal-Ezer. 2014.

[9] Wouter Groeneveld u. a. „Are Undergraduate Creative Coders Clean Coders?“ In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*. Hrsg. von Larry Merkle u. a. 2022.

[10] Felienne Hermans und Efthimia Aivaloglou. „Do code smells hamper novice programming? A controlled experiment on Scratch programs“. In: *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*. 2016.

[11] Felienne Hermans und Efthimia Aivaloglou. „Teaching Software Engineering Principles to K-12 Students: A MOOC on Scratch“. In: *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*. 2017.

[12] Dirk W. Hoffmann. *Software-Qualität*. Berlin, Heidelberg, 2013.

[13] Remin Kasahara u. a. „Applying Gamification to Motivate Students to Write High-Quality Code in Programming Assignments“. In: *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. Hrsg. von Bruce Scharlau u. a. 2019.

[14] Diana Kirk u. a. „High School Teachers' Understanding of Code Style“. In: *Koli Calling '20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research*. Hrsg. von Nick Falkner und Otto Seppala. 2020.

[15] Diana Kirk u. a. „On Assuring Learning About Code Quality“. In: *Proceedings of the Twenty-Second Australasian Computing Education Conference*. Hrsg. von Andrew Luxton-Reilly und Claudia Szabo. 2020.

[16] Diana Kirk u. a. „Teaching Code Quality in High School Programming Courses - Understanding Teachers' Needs“. In: *Australasian Computing Education Conference*. Hrsg. von Judy Sheard und Paul Denny. 2022.

[17] Philipp Mayring. *Qualitative Inhaltsanalyse: Grundlagen und Techniken*. 12., überarbeitete Auflage. Beltz, 2015.

[18] Annette Vee. *Coding literacy: How computer programming is changing writing*. Software studies. Cambridge, Massachusetts und London, England: The MIT Press, 2017.

[19] Helmut Witten. „Allgemeinbildender Informatikunterricht? Ein neuer Blick auf H. W. Heymanns Aufgaben allgemeinbildender Schulen“. In: *Informatische Fachkonzepte im Unterricht, INFOS 2003, 10. GI-Fachtagung Informatik und Schule*. Hrsg. von Peter Hubwieser. 2003.