HAW HAMBURG

BACHELORARBEIT

Einsatz von generativer KI in der Softwareentwicklung: Eine Systematische Literaturanalyse

vorgelegt am 04. Mai 2026
Michelle Koops

Erstprüfer: Dr. Jan Schwarzer
Zweitprüferin: Prof. Dr.-Ing. Sabine Schumann

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Department Medientechnik
Finkenau 35
20081 Hamburg

Zusammenfassung

Diese Bachelorarbeit untersucht mittels einer systematischen Literaturanalyse den Einsatz generativer Künstlicher Intelligenz (GenAI) in der Implementierungsphase der Softwareentwicklung. Ziel der Untersuchung ist es, den aktuellen Forschungsstand systematisch aufzubereiten und dabei sowohl den Zweck des Einsatzes als auch die damit verbundenen Risiken zu identifizieren.

Die Auswertung der identifizierten Primärstudien zeigt, dass GenAI weit über die reine Codeerzeugung hinaus als multifunktionales Assistenzsystem für Aufgaben wie Debugging, Refactoring und als interaktive Wissensquelle fungiert.

Die Ergebnisse verdeutlichen ein direktes Verhältnis zwischen Effizienzgewinnen und qualitativen Herausforderungen. Während die Automatisierung von Routineaufgaben zu einer kognitiven Entlastung führt, entstehen signifikante Risiken durch potenzielle Sicherheitsmängel im generierten Code und eine begrenzte Erfassung komplexer Projektzusammenhänge. Der daraus resultierende Validierungsaufwand kann die ursprüngliche Zeitersparnis teilweise wieder neutralisieren.

Abschließend wird dargelegt, dass die Integration von GenAI das Berufsbild der Softwareentwicklung transformiert. Die Tätigkeit verschiebt sich von der primär schreibenden Ausführung hin zu einer verstärkt prüfenden und steuernden Funktion. Der Mensch ist für den Erfolg dieser neuen Technologien weiterhin unerlässlich.

Abstract

This bachelor's thesis examines the use of generative artificial intelligence (GenAI) in the implementation phase of software development through a systematic literature review. The aim of the study is to systematically summarize the current state of research and to identify both the purpose of its use and the associated risks.

The evaluation of the identified primary studies shows that GenAI functions far beyond mere code generation as a multifunctional assistance system for tasks such as debugging, refactoring, and as an interactive knowledge source.

The results highlight a direct relationship between efficiency gains and qualitative challenges. While the automation of routine tasks leads to cognitive relief, significant risks arise from potential security flaws in the generated code and a limited grasp of complex project interdependencies. The resulting validation effort can partially offset the original time savings.

In conclusion, it is demonstrated that the integration of GenAI is changing the nature of the software development profession. The role is shifting from primarily writing code to an increasingly validating and controlling function. Humans remain indispensable for the success of these new technologies.

Inhalt

Abkürzungsverzeichnis

KI	Künstliche Intelligenz
GenAI	Generative Artificial Intelligence
LLM	Large Language Model
SLR	Systematic Literature Review
RQ	Research Question

1 Einleitung

1.1 Hintergrund und Motivation

Die Veröffentlichung von ChatGPT[1] im Dezember 2022 markierte einen Wendepunkt in der Wahrnehmung generativer Künstlicher Intelligenz (GenAI) (Wang, Ning, Zhang, Liu, & Wang, 2024).

Mit dem Aufkommen von Large Language Models (LLMs) erhielt Künstliche Intelligenz (KI) eine öffentliche Bekanntheit, die über das Interesse der Fachwelt hinausging und die öffentliche Aufmerksamkeit durch die Fähigkeit zur Generierung menschenähnlicher Texte, basierend auf natürlichen Spracheingaben (Prompts), auf sich zog. Die Nachfrage nach diesen Tools wird von der Größe der Nutzerbasis belegt, welche innerhalb von zwei Monaten nach der Veröffentlichung von ChatGPT 100 Millionen Menschen erreichte (Pereira, et al., 2025).

Diese Technologien haben einen großen Einfluss auf zahlreiche Felder, wobei besonders die Softwareentwicklung ein stark betroffenes Feld ist (Wang, Ning, Zhang, Liu, & Wang, 2024). Die Relevanz dieses Themas wird durch verschiedene Punkte verdeutlicht: Über 90 % der derzeit verfügbaren KI-Erweiterungen für integrierte Entwicklungsumgebungen (IDEs) wurden erst in den Jahren 2023 und 2024 veröffentlicht (Lyu, et al., 2025). Laut aktuellen Umfragen nutzen bereits über 76 % der Entwickler*innen KI-Tools oder planen deren Einsatz, um die eigene Arbeit zu verbessern (Pereira, et al., 2025).

Besonders in der Implementations- und Entwicklungsphase des Software-Engineering-Prozesses werden verschiedene KI-basierte Technologien (z. B. KI-Chatbots oder IDE-integrierte Assistenten) zunehmend zur Unterstützung von Programmieraufgaben eingesetzt (Khojah, Mohamad, Leitner, & De Oliveira Neto, 2024).

Diese Technologien ermöglichen es, Entwickler*innen bei Aufgaben wie der Codegenerierung, -vervollständigung, Fehleranalysen oder dem generellen Codeverständnis zu unterstützen (Wang, Ning, Zhang, Liu, & Wang, 2024). Zudem sorgt das Wegfallen bzw. die Automatisierung repetitiver Aufgaben wie das Schreiben von Boilerplate-Code – Code-Fragmente, die mit wenigen oder keinen Veränderungen immer wieder verwendet werden – oder ähnlicher, mühsamer Routineaufgaben für eine mentale Entlastung der Entwickler*innen. Das Erlernen neuer Kenntnisse oder das Ersetzen der klassischen Websuche trägt ebenfalls dazu bei. Dadurch ergeben sich mehr Möglichkeiten für das Bearbeiten kreativer und komplexerer Aufgaben, was zu mehr Motivation und Freude bei der Arbeit führt (Pereira, et al., 2025)

[1] https://chatgpt.com [abgerufen am 26.04.2026]

Ein Großteil der Studien deutet zudem auf Produktivitätsgewinne hin. In der Fachwelt führten die rasanten technologischen Fortschritte und Fähigkeiten generativer KI bereits zu Diskussionen über die Zukunft des Berufs des Programmierers (Pereira, et al., 2025) – doch trotz des schnellen Einzugs generativer KI in die Praxis deckt sich die tatsächliche Leistungssteigerung unter realen Umständen nicht immer mit den hohen Produktivitätsversprechen, da der Nutzen stark von der Komplexität der Aufgaben abhängt (Wang, Ning, Zhang, Liu, & Wang, 2024).

Der tatsächliche Nutzen der Integration von GenAI in die Entwicklung variiert stark je nach Art und Komplexität der Aufgabe oder Kenntnissen von Entwickler*innen (Baralla, Ibba, & Tonelli, 2025) und bringt auch verschiedene Risiken und Herausforderungen mit sich. Dies macht eine kritische Analyse der Ergebnisse unverzichtbar, insbesondere in Hinblick auf Sicherheitsrisiken durch Schwachstellen im generierten Code (Fu, et al., 2025). Hinzu kommen menschliche Risikofaktoren, wie die Neigung, KI-Vorschlägen aufgrund ihrer oberflächlichen Plausibilität zu vertrauen oder KI-generierte Lösungen ohne ausreichende manuelle Prüfung zu übernehmen (Vasconcelos, Bansal, Fourney, Liao, & Vaughan, 2024). Zudem drohen mögliche Kompetenzverluste durch abfallende Problemlösefähigkeiten sowie ein allgemeiner Qualitätsverlust durch fehlerhafte Logik (Pereira, et al., 2025).

1.2 Ziel der Arbeit

Ziel dieser Arbeit ist es, im Rahmen einer systematischen Literaturanalyse den aktuellen Forschungsstand zum Einsatz von generativer KI in der Implementierungsphase des Softwareentwicklungsprozesses strukturiert zusammenzuführen und anschließend zu diskutieren.

Dafür soll erfasst werden:

- für welche konkreten Zwecke GenAI innerhalb des Entwicklungsprozesses eingesetzt wird,
- welche objektiv messbaren oder subjektiv wahrgenommenen Nutzen für Entwickler*innen oder Unternehmen die Literatur dokumentiert,
- sowie welche Risiken mit der Nutzung von GenAI einhergehen; dies umfasst sowohl technische Mängel (z. B. Logikfehler) als auch menschliche Faktoren.

1.3 Forschungsfragen

Um die Zielsetzung dieser Arbeit zu erreichen, werden folgende Forschungsfragen (RQs) formuliert:

RQ1: Zu welchen konkreten Zwecken wird generative KI innerhalb der Implementierungsphase eingesetzt?

Diese Forschungsfrage dient dazu, die vielfältigen Einsatzszenarien von GenAI-Tools jenseits der bloßen Codeerstellung zu identifizieren. Hierbei soll untersucht werden, welche spezifischen Intentionen Entwickler*innen mit der Nutzung verfolgen. Die Frage zielt darauf ab, eine Übersicht der Rolle zu geben, die generative KI im Programmieralltag tatsächlich einnimmt.

RQ2: Von welchen messbaren (objektiven) oder wahrgenommenen (subjektiv) Nutzen sowie welchen Herausforderungen, Grenzen und Risiken wird in der Literatur im Zusammenhang mit der Integration von GenAI in den Entwicklungsprozess berichtet?

Ziel dieser Forschungsfrage ist die Darstellung der Auswirkungen des GenAI-Einsatzes auf etwa die individuelle Produktivität oder die resultierende Softwarequalität. Dabei wird zwischen objektiv quantifizierbaren Leistungsdaten und der subjektiven Wahrnehmung der Anwender*innen differenziert, um mögliche Diskrepanzen aufzuzeigen.

Ein weiterer Bestandteil der Untersuchung liegt zudem in der Erfassung bestehender Risiken, die sowohl technische als auch menschliche Faktoren adressieren. Durch diese Gegenüberstellung soll das Verhältnis zwischen den hohen Erwartungen an den technologischen Nutzen und den in der Praxis tatsächlich beobachtbaren Ergebnissen dargestellt werden.

1.4 Aufbau der Arbeit

Kapitel 1 behandelt die thematische Einführung sowie die Zielsetzung der Arbeit. Kapitel 2 beschreibt das methodische Vorgehen. Kapitel 3 präsentiert die Ergebnisse der Synthese, wobei Kapitel 4 die Ergebnisse diskutiert. Kapitel 5 schließt die Arbeit mit einer Zusammenfassung und einem Ausblick ab.

2 Methodik

Das folgende Kapitel beschreibt das methodische Vorgehen dieser Arbeit. Es orientiert sich an etablierten Standards für systematische Literaturanalysen in der Softwaretechnik und stellt die vorgenommenen Anpassungen an den Rahmen dieser Arbeit transparent dar. Schließlich werden die einzelnen Schritte des Vorgehens genauer beschrieben.

2.1 Systematische Literaturanalyse nach Kitchenham

Die Grundlage für das methodische Design bildet das von Barbara Kitchenham entwickelte Framework für Systematic Literature Reviews (SLR) (Kitchenham, 2004). Eine SLR ist eine Methode zur Identifizierung, Bewertung und Interpretation aller verfügbaren Forschungsergebnisse, die für eine bestimmte Forschungsfrage oder Themengebiet relevant sind. Im Gegensatz zu herkömmlichen Literaturübersichten zeichnet sich eine SLR durch ein vertrauenswürdiges und prüfbares Verfahren aus.

Nach Kitchenham gliedert sich dieser Prozess in drei wesentliche Phasen, die im Folgenden grob beschrieben sind:

1. **Planung**: In dieser Phase wird die Notwendigkeit der Analyse identifiziert und ein Review-Protokoll erstellt. Dieses ist entscheidend, um Forscher-Bias – die Tendenz, die eigene Forschung unbewusst durch eigene Überzeugungen zu beeinflussen – zu minimieren, da es die Forschungsfragen und Methoden für die Suche und Analyse bereits im Vorfeld festlegt.
2. **Durchführung**: Dieser Kernprozess umfasst fünf Schritte, darunter
    a. die Identifizierung der Forschung: Entwicklung einer umfassenden, unvoreingenommenen Suchstrategie,
    b. die Studienauswahl: Anwendung von Ein- und Ausschlusskriterien, die zuvor im Protokoll definiert wurden,
    c. die Qualitätsbewertung: Bewertung der Primärstudien hinsichtlich ihrer Validität und Qualität anhand spezifisch festgelegter Kriterien,
    d. die Datenextraktion: systematische Erfassung der benötigten Informationen aus den Studien mittels vordefinierter Formulare,
    e. die Datensynthese: Zusammenführung und Zusammenfassung der Ergebnisse, entweder deskriptiv oder quantitativ.
3. **Berichterstattung:** Die abschließende Phase befasst sich mit der strukturierten und zielgruppengerechten Darstellung der Ergebnisse.

2.2 Forschungsdesign und vorgenommene Anpassungen

Obwohl diese Arbeit den zentralen Prinzipien einer systematischen Literaturanalyse nach Kitchenham (2004) folgt, handelt es sich aufgrund des festgesetzten Bearbeitungszeitraums und der Durchführung durch eine einzelne Person nicht um eine vollumfängliche SLR im Sinne Kitchenhams. Ein vollständiges SLR-Verfahren ist ressourcenintensiv und sieht idealerweise die Zusammenarbeit mehrerer Forscher vor, um Objektivität zu garantieren. Stattdessen wurde ein angepasstes Vorgehen gewählt, das bezüglich des Umfangs und der Tiefe einzelner Schritte reduziert ist.

Um die Durchführbarkeit im Rahmen einer Bachelorarbeit zu gewährleisten, wurden folgende Anpassungen vorgenommen, welche in der Diskussion als Limitationen berücksichtigt werden:

- **Review-Protokoll:** Ein zentrales Element der Planungsphase nach Kitchenham ist die Erstellung eines Review-Protokolls, welches idealerweise von unabhängigen Dritten validiert wird, um die Objektivität des Verfahrens sicherzustellen. In dieser Arbeit wurde auf die Erstellung eines separaten Dokuments und – aufgrund der Durchführung als Einzelprojekt – auf die externe Prüfung des Protokolls verzichtet. Dennoch wurde für die systematische Nachvollziehbarkeit eine Suchstrategie ausgearbeitet, sodass insgesamt die wichtigsten Aspekte eines formalen Review-Protokolls abgedeckt sind.
- **Eingeschränkte Quellenvielfalt & Suchstrategie:** Während Kitchenham eine möglichst umfassende Suche in zahlreichen Quellen (Datenbanken, Handsuche, graue Literatur) anstrebt, beschränkt sich diese Arbeit auf zwei wissenschaftliche Datenbanken sowie einen eng gefassten Suchstring. Dies dient dazu, eine im gegebenen Zeitrahmen handhabbare Treffermenge von Studien zu erzielen.
- **Keine doppelte Prüfung** (Vier-Augen-Prinzip): In einem Standard-SLR sollten die Auswahl der Studien und die Datenextraktion unabhängig von mindestens zwei Gutachtern durchgeführt und verglichen werden. Da diese Analyse allein durchgeführt wurde, besteht ein höheres Risiko für subjektive Verzerrungen.
- **Verzicht auf formale Qualitätsbewertung:** Kitchenham sieht die Verwendung von "Quality Instruments" (Checklisten) vor, um die methodische Stärke und Validität jeder Studie zu gewichten. In dieser Arbeit wurde auf eine tiefgehende, kriteriengestützte Bewertung verzichtet; stattdessen bildet die Erfüllung der in Kapitel 2.4 formulierten Ein- und Ausschlusskriterien den Mindestqualitätsstandard, um die inhaltliche Relevanz der Literatur sicherzustellen.

- **Vereinfachte Synthesemethode**: Im Rahmen dieser Arbeit wird der Prozess der Synthese gezielt vereinfacht umgesetzt. Anstatt komplexer qualitativer Verfahren konzentriert sich die Synthese auf den systematischen Vergleich der extrahierten Merkmale in tabellarischer Form.

2.3 Suchstrategie

Zur Identifikation relevanter Primärstudien wurde eine zielgerichtete Vorgehensweise, basierend auf den Aspekten eines Review-Protokolls nach Kitchenham (2004), für die Literaturrecherche erarbeitet. Diese umfasste neben der Formulierung der in Kapitel 1.3 genannten Forschungsfragen auch die Auswahl geeigneter Datenbanken, das Entwickeln spezifischer Suchstrings und die Definition von Ein- und Ausschlusskriterien.

Die einzelnen Bestandteile der Suchstrategie werden in den folgenden Abschnitten genauer beschrieben.

2.3.1 Datenbanken

Die Literatursuche wurde in folgenden Datenbanken durchgeführt:

- **ACM Digital Library**
- **IEEE Xplore**

Angewandte Filter:

Zeitraum: 2024 - 2026

Publikationstyp: Journal-Artikel und Konferenzbeiträge

Die angewandten Filter dienten erneut der gezielten Eingrenzung der Treffermenge. Der Suchzeitraum wurde auf die Jahre 2024 bis 2026 beschränkt, um möglichst aktuelle Entwicklungen des Forschungsfeldes abzubilden. Zudem wurden ausschließlich Journal-Artikel und Konferenzbeiträge berücksichtigt, da diese als wissenschaftlich relevante und qualitätsgesicherte Publikationsformate gelten.

2.3.2 Suchstrings

Für die Suche wurde folgende Suchanfrage verwendet:

((„genAI“ OR „generative ai“ OR „artificial intelligence“) AND „software development“ AND „tool?“ AND „developer?“ AND „productivity“ AND „code“)

Die Suchanfrage wurde iterativ entwickelt und so gestaltet, dass ein möglichst direkter Bezug zur Nutzung generativer KI im Softwareentwicklungsprozess hergestellt werden kann.

- „genAI“ / „generative ai“ / „artificial intelligence”: Diese Begriffe decken das gesamte technologische Spektrum ab. Während „artificial intelligence“ als notwendiger Oberbegriff dient, um auch Studien zu erfassen, die modernere Ansätze noch unter diesem Begriff führen, dienen „genAI“ und „generative ai“ der zeitlichen Einordnung und stellen den direkten Bezug zur aktuellen KI-Technik (insbesondere Large Language Models) her.

- „software development”: Dieser Begriff dient der Einschränkung des Anwendungskontexts, um die Suche strikt auf den Bereich der Softwareentwicklung zu begrenzen, sodass allgemeine KI-Studien aus anderen Fachbereichen (z. B. Medizin) ausgeschlossen werden. Zudem wurde bewusst auf den Oberbegriff „software engineering“ verzichtet, da der Fokus dieser Arbeit explizit auf der Implementierungsphase liegt – durch diese spezifischere Wahl konnten schon vorab manche Studien, die sich mit anderen Phasen des Software-Lebenszyklus befassen (z. B. Anforderungsmanagement oder reines Projektmanagement), ebenfalls ausgeschlossen werden.

- „tool?“ / „developer?“: Diese Begriffe dienen der gezielten Eingrenzung auf Studien, die den Einsatz KI-gestützter Systeme im softwareentwicklungsbezogenen Arbeitskontext sowie deren Nutzung durch Entwickler*innen thematisieren. Während „tool?“ die Suche auf die praktische Anwendung technischer Assistenzsysteme (z. B. IDE-Integrationen wie GitHub Copilot[2]) fokussiert, stellt „developer?“ sicher, dass Studien identifiziert werden, welche die Interaktion zwischen Mensch und KI sowie die Nutzerperspektive von Softwareentwickler*innen untersuchen.

- „productivity“: Dieser Begriff zielt direkt auf eine der Forschungsfragen (RQ2) ab. Er stellt den direkten Bezug zu den untersuchten Metriken wie Effizienz oder Zeitersparnis her. In der Suchstrategie dient er zur Ergebnisreduzierung, da er allgemeine Berichte über KI-Technologien ausschließt, die keine Aussagen über deren tatsächlichen Nutzen oder Auswirkungen auf den Arbeitsfluss treffen.

- „code“: Da die Arbeit die Implementierungsphase untersucht, dient dieser Begriff als spezifischer technischer Filter und stellt sicher, dass die identifizierten Studien einen direkten Bezug zur praktischen Programmierung aufweisen. Er dient außerdem gezielt dazu, die Treffermenge weiter zu minimieren.

[2] https://github.com/copilot [abgerufen am 26.04.2026]

Die Kombination dieser Suchbegriffe konnte die initiale Treffermenge auf eine geringere Menge reduzieren, um eine gründliche Analyse der Literatur innerhalb des vorgegebenen Zeitrahmens zu ermöglichen. Zur Validierung des Suchstrings wurde stichprobenartig geprüft, ob bekannte relevante Studien durch die Suchanfrage weiterhin gefunden werden konnten.

2.4 Ein- und Ausschlusskriterien

Die folgenden Kriterien wurden definiert und bildeten die Grundlage für den mehrstufigen Studienauswahlprozess.

2.4.1 Einschlusskriterien

Um mit in die Analyse aufgenommen zu werden, mussten folgende Bedingungen erfüllt werden:

- **Typ**: Es handelt sich um eine Primärstudie, die auf einer eigenen Datenerhebung oder -auswertung basiert.
- **Fokus auf die Implementierungsphase**: Die Untersuchung muss sich explizit auf Tätigkeiten innerhalb der Entwicklungsphase des Software-Engineering-Prozesses beziehen.
- **Nutzung durch Entwickler*innen**: Die Studie muss die Interaktion zwischen menschlichen Entwickler*innen und generativen KI-Tools thematisieren.
- **Bezug zu Forschungsfragen:** Die Publikation muss relevante Informationen zu mindestens einem der Kernaspekte der Forschungsfragen liefern. Dazu zählen die Identifikation konkreter Einsatzszenarien, die Dokumentation eines messbaren oder wahrgenommenen Nutzens oder die Analyse von Herausforderungen und Risiken.
- **Dokumentation der Ergebnisse**: Die Studie muss ihre Ergebnisse in einer strukturierten und ausreichend detaillierten Form darstellen, sodass zentrale Erkenntnisse für die Analyse verwertbar sind.

2.4.2 Ausschlusskriterien

Publikationen wurden von der Datenextraktion ausgeschlossen, wenn eines der folgenden Kriterien zutraf:

- **Sprache**: Es handelt sich um eine Publikation, die nicht in englischer Sprache verfasst ist.

- **Kontext**: Die Arbeit bezieht sich auf einen rein akademischen Kontext ohne industriellen Bezug.
- **Typ**: Es handelt sich um konzeptionelle oder visionäre Beiträge, rein theoretische Positionspapiere, Meinungsbeiträge oder Roadmaps, die lediglich Potenziale aufzeigen.
- **Abweichender Fokus**: Die Arbeit hat einen rein technischen Schwerpunkt (z. B. Modellarchitektur oder Trainingsverfahren) ohne Bezug zur praktischen Nutzung durch Entwickler*innen.
- **Duplikat:** Identische Treffer (Duplikate), die in mehreren Datenbanken gefunden wurden, werden auf eine einzige Primärquelle reduziert.

2.5 Studienauswahlprozess

Der Auswahlprozess der Primärliteratur wurde als mehrstufiges Verfahren gestaltet, um die hohe Treffermenge systematisch auf die für die Forschungsfragen relevantesten Arbeiten zu reduzieren. Die Auswahl erfolgte in vier Phasen, die anschließend in Abbildung 1 visualisiert wurden.

1. **Datenbanksuche**: Die Suche in den wissenschaftlichen Datenbanken unter Verwendung des in Suchstrings 2.3.2 definierten Suchstrings ergab eine Gesamtmenge von 127 potenziellen Studien (ACM: 80 Treffer, IEEE: 47 Treffer). Dies bildete die Ausgangsbasis für den weiteren Screening-Prozess.
2. **Titel-Prüfung:** Die zweite Phase stellt eine grobe Relevanzprüfung dar. In dieser wurden die Titel auf Relevanz geprüft. Studien, deren Titel einen potenziellen Bezug zur Softwareentwicklung aufwiesen, wurden zunächst beibehalten. Ausgeschlossen wurden hingegen Arbeiten, die bereits im Titel eine klare thematische Abgrenzung aufwiesen, etwa durch expliziten Fokus auf andere Phasen des Software-Lebenszyklus (z. B. Testing, Anforderungsmanagement) oder rein hardwarebezogene Themen. Es verblieben 52 Studien.
3. **Abstract-Prüfung**: Die verbliebenen Arbeiten wurden durch das Lesen der Abstracts geprüft. Erstens wurde sichergestellt, dass es sich tatsächlich um Primärstudien handelt, die empirischen Daten durch z. B. Experimente oder Fallstudien erheben. Zweitens wurde geprüft, ob ein Bezug zu den Forschungsfragen dieser Arbeit hergestellt werden kann. Die Anzahl verringerte sich auf 18 Studien.
4. **Volltext-Prüfung:** Im letzten Schritt wurde der vollständige Text der Arbeiten betrachtet. Dabei wurde die inhaltliche Relevanz dahingehend bewertet, ob die Studie explizit

die Implementierungsphase adressiert und konkrete Aussagen zu den Forschungsfragen trifft. Ebenso wurde die empirische Tiefe kritisch geprüft, indem sichergestellt wurde, dass die Arbeiten nicht nur oberflächliche Beobachtungen teilen, sondern belastbare und extrahierbare Daten liefern. Dies umfasst ebenso die methodische Transparenz.

Unter Berücksichtigung der zeitlichen Limitationen der Arbeit wurden so 13 Studien für die spätere Synthese ausgewählt.

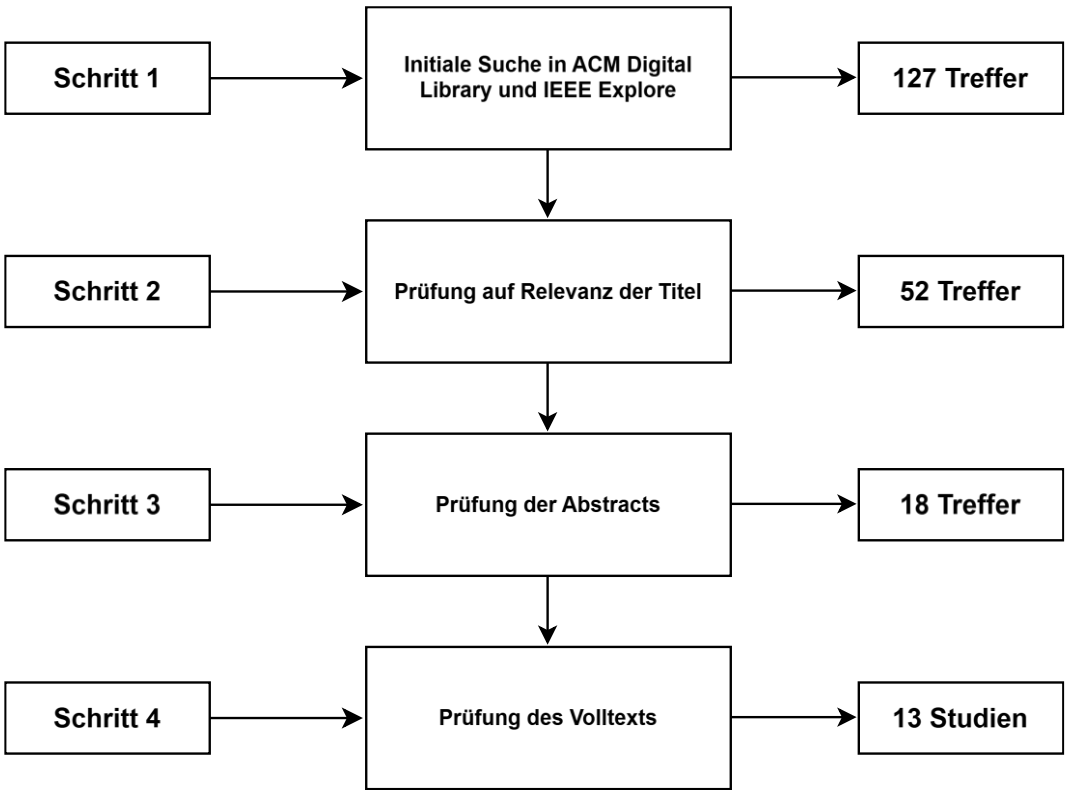

Abbildung 1: Studienauswahlprozess nach Dybå & Dingsøyr (2008)

2.6 Datenextraktion

Nach der finalen Auswahl der Literatur erfolgte die Extraktion spezifischer Informationen, um eine Übersicht der empirischen Erkenntnisse für die anschließende Synthese zu schaffen.

2.6.1 Extrahierte Merkmale

Die extrahierten Merkmale umfassen die folgenden Kategorien:

- **Allgemeine Publikationsdaten:** Erfasst wurden der Titel, die Autor*innen, das Publikationsjahr bzw. -datum sowie der Publikationstyp. Jeder Studie wurde eine Identifikationsnummer zugewiesen.

- **Studiendesign und Rahmenbedingungen:** Um später eventuelle methodische Limitationen einzuordnen, wurden der Forschungstyp (z. B. randomisierte kontrollierte Studie (RCT), Tagebuchstudie oder Fallstudie), die Stichprobengröße (N), sowie die Dauer der Untersuchung, sofern vorhanden, dokumentiert. Dies ermöglicht unter anderem die Unterscheidung zwischen kurzfristigen Effekten in kontrollierten Umgebungen und Ergebnissen aus dem realen Arbeitsalltag über mehrere Wochen hinweg. Zusätzlich wurde erfasst, welche spezifischen KI-Modelle oder Werkzeuge – beispielsweise GPT-4, GitHub Copilot oder Amazon CodeWhisperer, dessen Funktionen heute Teil von Amazon Q Developer[3] sind – Teil der Untersuchungen waren. Um die Validität der Ergebnisse zu bewerten, wurden schließlich die in der Literatur genannten Einschränkungen notiert.
- **Inhaltliche Schwerpunkte bezüglich der Forschungsfragen:** Neben den Kernaussagen der Studie wurden konkrete Aussagen in Bezug auf die in Kapitel 1.3 formulierten RQs extrahiert. Dazu zählen Informationen zu den Zwecken und Aufgabenbereichen (RQ1) sowie die dokumentierten Nutzenformen, Risiken und Herausforderungen (RQ2)

2.6.2 Synthesemethode

Nach dem Framework von Kitchenham lassen sich Synthesemethoden grundsätzlich in quantitative (Meta-Analysen) und deskriptive Ansätze unterteilen. Eine quantitative Synthese erfordert eine hohe Homogenität (Vergleichbarkeit) der Daten, um statistische Effekte berechnen zu können (Kitchenham, 2004). Da die in dieser Arbeit identifizierten 13 Primärstudien methodisch sehr heterogen sind – das Spektrum reicht von RCTs über Fallstudien bis hin zu Tagebuchstudien und Marktplatz-Analysen – wurde ein deskriptiver Syntheseansatz gewählt.

Die Synthese konzentriert sich im ersten Schritt auf den systematischen Vergleich der im vorigen Abschnitt beschriebenen extrahierten Merkmale. Tabelle 1 stellt die dafür genutzte Datenextraktionstabelle beispielhaft dar.

Diese strukturierte Zusammenführung ermöglicht es, Ähnlichkeiten und Widersprüche in den Ergebnissen der Primärstudien sowie unterschiedliche Forschungsansätze gegenüberzustellen.

Im zweiten Schritt wurden die Erkenntnisse bezüglich der Forschungsfragen gruppiert, um später eine Beantwortung dieser Fagen zu ermöglichen:

[3] https://aws.amazon.com/q/developer [abgerufen am 26.04.2026]

- Bezug zu RQ1: Die Einsatzbereiche werden nach Zweck und Aufgabentypen kategorisiert (z. B. Codegenerierung, Debugging, Refactoring).
- Bezug zu RQ2: Die berichteten Effekte werden bezüglich ihres Nutzens und ihres Risikos gegenübergestellt, wobei zusätzlich zwischen objektiven Messdaten und subjektiver Wahrnehmung unterschieden wird.

Studienbeschreibung	
Studien-ID	Eindeutige Identifikationsnummer für die Studie
Allgemeine Publikationsdaten	Titel, Autor*innen, Publikationsdatum/-jahr, Quelle
Publikationstyp	Journal-Artikel, Konferenzbeitrag
Forschungsdesign	z. B. Fallstudie, RCTs, Umfrage
Stichprobenbeschreibung	Größe der Stichprobe, Beschreibung der Teilnehmer oder untersuchten Artefakte
Forschungsdauer	Zeitlicher Rahmen oder Zeitpunkt der jeweiligen Untersuchung
Studienergebnisse	
Kernaussagen	Zusammenfassung der Hauptergebnisse aus der Studie
Bezüge zu Forschungsfragen	Für welche Bereiche der Softwareentwicklung und welche Zwecke wurde GenAI eingesetzt? Welche daraus entstehenden Nutzen wurden dokumentiert? Welche Herausforderungen und Risiken wurden genannt?
Validität	Einschränkungen der Validität laut Autor*innen

Tabelle 1: Datenextraktionstabelle nach Dybå & Dingsøyr (2008)

3 Synthese

Im folgenden Abschnitt werden die Ergebnisse aus den Studien hinsichtlich der in Kapitel 1.3 formulierten Forschungsfragen analysiert.

3.1 Überblick über die identifizierten Studien

Die Literatur umfasst ein breites Spektrum an Forschungstypen, darunter beispielsweise kontrollierte Experimente (Paradis, et al., 2025), Umfragen unter professionellen Entwickler*innen (Pereira, et al., 2025), Tagebuchstudien (Butler, Suh, Haniyur, & Hadley, 2025) und systematische Analysen von Marktplatz-Reviews (Lyu, et al., 2025).

Technologisch stehen überwiegend Chatbots (wie ChatGPT, Google Gemini[4]) oder IDE-Integrationen (wie GitHub Copilot, Amazon CodeWhisperer) im Fokus. Seltener sind Plug-ins wie *CodeBlizz*, die neben der Codegenerierung auf die Bereitstellung von Lernressourcen und Echtzeit-Debugging innerhalb von Visual Studio Code[5] setzen (Preethi, Ragavan, Abinandhana, Umamaheswari, & Suvethaa, 2024).

Tabelle 2 stellt eine knappe Übersicht der Studien dar.

ID	Forschungstyp	Forschungs-dauer oder Zeitpunkt	Stichprobenbeschrei-bung	Untersuchte Tools
S1	Empirische Evalua-tion (Experiment)	2024	143 Smart Contracts[6]	GitHub Copilot
S2	RCT, Tagebuchstudie	3 Wochen	106 Softwareentwick-ler*innen	GitHub Copilot
S3	Empirische Studie (Statische Sicher-heitsanalyse)	-	733 Code-Schnipsel	Copilot[7], Ama-zon CodeWhis-perer, Codeium[8]
S4	Beobachtungsstudie	1 Woche	24 Softwareentwick-ler*innen	ChatGPT (GPT-3.5)

[4] https://gemini.google.com [abgerufen am 26.04.2026]
[5] https://code.visualstudio.com [abgerufen am 28.04.2026]
[6] Digitale Verträge, die auf einer Blockchain gespeichert werden, um Vereinbarungen bei Erfüllung bestimmter Bedingungen automatisch auszuführen.
[7] https://copilot.microsoft.com [abgerufen am 26.04.2026]
[8] https://codeium.en.softonic.com [abgerufen am 26.04.2026]

S5	Empirische Studie (Review Mining)	-	361 Nutzer-Reviews	32 Visual Studio Code Erweiterungen
S6	RCT	30 Minuten–2,5 Stunden pro Aufgabe	96 Softwareentwickler*innen	Interne Google-KI-Tools
S7	Fallstudie	6 Wochen	66 Teilnehmer (Erstumfrage), 23 Teilnehmer (Zweitumfrage)	Google Gemini
S8	Tool-Entwicklung, Umfrage	-	-	CodeBlizz Plugin (GPT-basiert)
S9	Fallstudie, Umfrage	2024	-	ChatGPT, Google Gemini, Copilot
S10	Nutzerstudie	-	30 Programmierer*innen	OpenAI Codex[9]
S11	Laborexperiment	75 Minuten pro Aufgabe	109 Teilnehmer*innen	ChatGPT (GPT-3.5)
S12	Nutzerstudie	15 Minuten pro Aufgabe	24 Softwareentwickler*innen	Copilot
S13	Benchmarking	-	5 LLMs, 164 Probleme	CodeLlama[10], DeepSeek Coder[11], DeepSeek Chat[12], CodeQwen1.5[13], ChatGPT

Tabelle 2: Übersicht der identifizierten Studien

[9] https://openai.com/codex [abgerufen am 26.04.2026]
[10] https://ollama.com/library/codellama [abgerufen am 26.04.2026]
[11] https://deepseekcoder.github.io [abgerufen am 26.04.2026]
[12] https://chat.deepseek.com [abgerufen am 26.04.2026]
[13] https://qwen.ai [abgerufen am 26.04.2026]

3.2 Zwecke und Einsatzszenarien

Generative KI wird in der Entwicklungsphase für ein breites Spektrum an Aufgaben eingesetzt. In diesem Abschnitt werden die vielfältigen Einsatzszenarien und die funktionale Rolle der generativen KI dargestellt.

Tabelle 3 veranschaulicht knapp die Häufigkeit, mit der die verschiedenen Einsatzszenarien in der identifizierten Literatur untersucht wurden.

Zweck, Einsatzszenario	**Häufigkeit**
Codegenerierung und -vervollständigung	13/13
Fehlersuche, Debugging & Bugfixing	12/13
Refactoring & Code-Optimierung	5/13
Lern- und Wissensquelle	8/13
Kognitive Entlastung	6/13

Tabelle 3: Häufigkeiten der Einsatzszenarien

3.2.1 Codegenerierung und -vervollständigung

Die Erzeugung und Vervollständigung von Programmcode stellt in der untersuchten Primärliteratur oft eines der häufigsten Einsatzgebiete dar (Pereira, et al., 2025). Die Technologie wird dabei primär in zwei verschiedenen Interaktionsmodi eingesetzt: der Echtzeit-Vervollständigung (Auto-complete) in z. B. IDE-Integrationen und der Generierung aus natürlichen Sprachbefehlen (Prompts) (Weber, Brandmaier, Schmidt, & Mayer, 2024).

3.2.2 Fehlersuche, Debugging & Bugfixing

Ein weiterer wichtiger und häufig genannter Aspekt ist die Fehlerbehebung (Debugging). GenAI wird zur Identifizierung von Fehlern, zur Analyse von Fehlermeldungen und zur Generierung von Korrekturvorschlägen eingesetzt (Butler, Suh, Haniyur, & Hadley, 2025). In einer Umfrage von Urek et al. (2025) gaben 48,2 % der professionellen Entwickler*innen an, KI-Tools speziell für das Bugfixing zu nutzen. Weitere Studien belegen, dass Entwickler*innen simple Bugs mit der KI-Unterstützung etwa 50 % schneller beheben konnten als mit manuellen Methoden (Preethi, Ragavan, Abinandhana, Umamaheswari, & Suvethaa, 2024)

3.2.3 Refactoring & Code-Optimierung

Die analysierten Studien berichteten ebenso vom Einsatz generativer KI bei der Refaktorisierung (Refactoring) und Optimierung des Codes. Dieser Bereich ist von hoher Bedeutung, da in realen Projektumgebungen (z. B. auf GitHub[14]) 81 % der Aktivitäten auf der Modifikation bestehenden Codes beruhen (Wang, Ning, Zhang, Liu, & Wang, 2024)

3.2.4 Lern- und Wissensquelle

Gleichzeitig gewinnt die Nutzung als Lern- und Wissensquelle an Bedeutung. Entwickler*innen setzen Werkzeuge wie ChatGPT oder Copilot zunehmend statt der traditionellen Websuche (z. B. Google) oder für Portale wie Stack Overflow[15] (Butler, Suh, Haniyur, & Hadley, 2025) ein. Sie suchen dort nach Codebeispielen, Erklärungen für komplexe Logik (Urek, Maslać, & Budin, 2025) oder Hilfe bei der Syntax unbekannter Programmiersprachen zu erhalten (Butler, Suh, Haniyur, & Hadley, 2025).

Dieses Muster tritt besonders in den längerfristigen Studien und Projekten unter realen Arbeitsbedingungen (Butler, Suh, Haniyur, & Hadley, 2025) auf, während es in kontrollierten Experimenten von kurzer Dauer (Wang, Ning, Zhang, Liu, & Wang, 2024) weniger stark ausgeprägt ist.

Mehrere Studien beschreiben GenAI außerdem als eine Art „virtuellen Kollegen“ (Wang, Ning, Zhang, Liu, & Wang, 2024). In dieser Rolle wird die KI für eine Art Brainstorming von Programmierideen oder das Durchspielen von Problemlösungsansätzen genutzt. Dabei dient die KI auch oft als initialer Startpunkt für neue Projekte oder neue Funktionen, der dann manuell verfeinert wird (Khojah, Mohamad, Leitner, & De Oliveira Neto, 2024).

3.2.5 Kognitive Entlastung

Ein weiterer zentraler Nutzen ist die Entlastung bei repetitiven Tätigkeiten. Hierzu zählt beispielsweise das Schreiben von Boilerplate-Code, Imports von Paketen oder sonstigen einfachen Tätigkeiten, die als alltägliche Routineaufgaben angesehen werden (Lyu, et al., 2025). Die KI wird hier als effizientes Werkzeug wahrgenommen, um mühsame Arbeit zu reduzieren und den Fokus auf komplexere Probleme zu ermöglichen (Pereira, et al., 2025).

[14] https://github.com [abgerufen am 26.04.2026]
[15] https://stackoverflow.com/ [abgerufen am 26.04.2026]

3.3 Nutzenpotenziale, Herausforderungen und Risiken

Der folgende Abschnitt befasst sich mit dem Verhältnis zwischen dem in der Literatur dokumentierten Nutzen der GenAI-Integration und den damit einhergehenden Herausforderungen und Risiken.

Tabelle 4 stellt knapp die Häufigkeit dieser dokumentierten Nutzenpotenziale sowie Risiken und Herausforderungen dar.

Nutzen, Risiken oder Herausforderungen	**Häufigkeit**
Produktivitätssteigerung	12/13
Reduzierung des manuellen Aufwands	8/13
Lernunterstützung	7/13
Wahrgenommener vs. tatsächlicher Nutzen	6/13
Selbstwahrnehmung (Motivation/Arbeitsfreude)	4/13
Sicherheitslücken im Code	5/13
Problem der Komplexität	5/13
Kommunikationsdefizit & Halluzinationen	3/13
Kompetenzabfall & Abhängigkeit	3/13
Automation Bias & Overreliance	3/13
Qualitätsabfall	3/13
Validierungsaufwand	5/13
Datenschutz & Ethische Bedenken	4/13

Tabelle 4: Häufigkeiten der Nutzenpotenziale, Risiken, Herausforderungen

3.3.1 Produktivitätssteigerung

Kontrollierte Laborstudien auf Basis kleiner Programmieraufgaben berichten von hohen Produktivitätssteigerungen: Wang, Ning, Zhang, Liu, & Wang (2024) stellen eine signifikante Effizienzsteigerung bei einfachen Programmierrätseln fest und Weber, Brandmaier, Schmidt, & Mayer (2024) dokumentieren eine Zunahme der implementierten Anforderungen um 65 % durch den Einsatz von KI-Assistenten.

Die Messungen in realen Unternehmensumgebungen fallen dagegen jedoch moderater aus. Eine großangelegte Studie bei Google benennt die tatsächliche Zeitersparnis bei realen Entwicklungsaufgaben mit etwa 21 % bis 26 % (Paradis, et al., 2025). In einem ähnlichen Bereich bewegen sich die Ergebnisse einer weiteren Fallstudie aus der Industrie, die von einer Reduktion der (Entwicklungs-)Zykluszeit (Cycle Time) um 23 % berichten (Pereira, et al., 2025).

3.3.2 Reduzierung des manuellen Aufwands

Dieser Nutzen resultiert vor allem aus der Reduzierung manuellen Aufwands bei den in 3.2 genannten Routineaufgaben. Dies führt zu einer kognitiven Entlastung: 78,6 % der befragten Entwickler*innen einer Studie geben an, dass GenAI den mentalen Aufwand für repetitive (Routine-)Aufgaben für sie reduziert (Pereira, et al., 2025), wodurch sich Entwickler*innen stärker auf anspruchsvollere logische Probleme konzentrieren können (Preethi, Ragavan, Abinandhana, Umamaheswari, & Suvethaa, 2024).

3.3.3 Lernunterstützung

Zudem fungieren die Tools als starke **Lernunterstützung**: 61,2 % der Befragten einer Studie geben an, durch die KI-Vorschläge neue Konzepte oder Programmiersprachen schneller zu erlernen (Pereira, et al., 2025).

Vor allem für Junior-Entwickler*innen bestehen große Potenziale für die Wissensbeschaffung. Das spezialisierte Plugin *CodeBlizz* wurde beispielsweise gezielt entwickelt, Lernressourcen direkt in die Entwicklungsumgebung zu integrieren, was in Nutzertests zu einer Steigerung des Selbstvertrauens und einer Senkung der Frustration führte (Preethi, Ragavan, Abinandhana, Umamaheswari, & Suvethaa, 2024).

3.3.4 Wahrgenommener vs. tatsächlicher Nutzen

In der Literatur zeigt sich jedoch ein Muster einer Diskrepanz zwischen dem wahrgenommenen und dem tatsächlich messbaren Nutzen. Etwa 84 % der Entwickler*innen einer Studie berichten subjektiv von positiven Veränderungen ihrer Arbeitspraktiken und einem gesteigerten Produktivitätsgefühl.

Demgegenüber stehen jedoch objektive Messdaten (wie die Anzahl der Pull-Requests oder geschriebenen Codezeilen), die keine großen Unterschiede zwischen KI-Nutzern und der Kontrollgruppe ohne Zugang zu KI feststellen konnten (Butler, Suh, Haniyur, & Hadley, 2025).

3.3.5 Selbstwahrnehmung der Anwender*innen

Wie bereits angedeutet zeigt sich ein überwiegend positives Bild hinsichtlich der Selbstwahrnehmung der eigenen Arbeit und der emotionalen Einstellung von Entwickler*innen gegenüber generativer KI. Während weiter oben genannte Ergebnisse teils variieren, ist die individuelle Zufriedenheit und der wahrgenommene Nutzen konsistent hoch.

Ein Großteil der Anwender*innen empfindet die Erfahrung mit GenAI als sehr zufriedenstellend (Pereira, et al., 2025). Die Nutzung dieser Tools führt bei vielen Entwickler*innen zu einer gesteigerten Freude an der Arbeit bzw. Programmierung, da mühsame Routineaufgaben wegfallen und die Arbeit als weniger langweilig empfunden wird (Butler, Suh, Haniyur, & Hadley, 2025). Diese positive Einschätzung ist bei Junior-Entwickler*innen, die die Tools fast ausnahmslos als hilfreich beim Projektbau empfinden (14 von 15 in einer Stichprobe), deutlich stärker ausgeprägt als bei erfahrenen Entwickler*innen, die den Nutzen aufgrund der notwendigen Validierung skeptischer bewerten (Lyu, et al., 2025).

Während über die Hälfte der Befragten (55,3 %) einer Studie davon ausgeht, dass KI ihre Rolle als Entwickler*innen zwar verändern, aber nicht ersetzen wird, sondern sie produktiver macht, lassen sich weiterhin Sorgen hinsichtlich der langfristigen beruflichen Sicherheit in der Literatur identifizieren. Fachkräfte rechnen mit einer starken Veränderung ihres Berufsalltags und befürchten, dass KI-gestützte Effizienzgewinne als Rechtfertigung für Kosteneinsparungen, Stellenstreichungen oder eine Erhöhung der Arbeitsbelastung herangezogen werden könnten (Urek, Maslać, & Budin, 2025).

3.3.6 Problem der Komplexität

Ein weiteres Hindernis sind die Grenzen bei steigender Komplexität der Aufgabenstellungen. Die Leistungsfähigkeit von KI-Tools sinkt drastisch ab, sobald Aufgabenstellung in Form einer Programmieraufgabe (z. B. Algorithmus) überstiegen wird (Wang, Ning, Zhang, Liu, & Wang, 2024) und stattdessen komplexe Logik oder ein projektweiter Kontext erforderlich sind, den die Modelle aufgrund mangelnden Kontextbewusstseins nicht vollständig erfassen können (Baralla, Ibba, & Tonelli, 2025).

So haben KI-Assistenten beispielsweise Schwierigkeiten, Definitionen in Dateien zu berücksichtigen, die nicht aktuell geöffnet sind, oder komplexe Abhängigkeiten in großen Codebasen wie bei realen Entwicklungsprojekten korrekt zu verarbeiten (Wang, Ning, Zhang, Liu, & Wang, 2024).

3.3.7 Sicherheitslücken im Code

Eng verbunden mit der Komplexität ist die dabei entstehende Unzuverlässigkeit (Baralla, Ibba, & Tonelli, 2025), vor allem bezüglich kritischer Aspekte wie Sicherheitslücken: Empirische Analysen von KI-generierten Code-Ausschnitten in Open-Source-Projekten auf GitHub zeigen, dass etwa 27,3 % bis 30 % der untersuchten Code-Ausschnitte Sicherheitsmängel aufweisen. Bei einer Studie, die sich explizit mit der Sicherheit beschäftigte, treten Schwachstellen am häufigsten bei unzureichenden Zufallswerten, Code-Injektion und webseitenübergreifendem Skripting (XSS; Cross-Site Scripting) auf (Fu, et al., 2025).

3.3.8 Kommunikationsdefizit & Halluzinationen

Technische Defizite können durch ein Kommunikationsdefizit verschärft werden, das oft als „Raten statt Fragen“ beschrieben wird. Die Ergebnisse von Wu & Fard (2025) zeigen, dass Large Language Models (LLMs) in über 60 % der Fälle direkt Antworten generieren, selbst wenn die Anforderungen offensichtlich unvollständig, widersprüchlich oder mehrdeutig formuliert sind, anstatt klärende Rückfragen zu stellen. Erst wenn ein Großteil der Problembeschreibung (bis zu 90 %) fehlt, steigt die Bereitschaft der KI, Fragen zu stellen, auf etwa 54 %.

Dieses Kommunikationsdefizit ist zudem oft die Ursache für eine Halluzination. Darunter wird das Erzeugen von Texten verstanden, die zwar oberflächlich betrachtet plausibel und korrekt klingen, jedoch faktisch falsch oder unsinnig sind. Dieser Fehler tritt vor allem dann auf, wenn die Trainingsdaten der KI nicht ausreichen, um eine spezifische Anforderung korrekt zu erfüllen (Khojah, Mohamad, Leitner, & De Oliveira Neto, 2024).

3.3.9 Qualitätsabfall

Die oben genannten Punkte weisen trotz der Effizienzgewinne (Wang, Ning, Zhang, Liu, & Wang, 2024) auf einen potenziellen Qualitätsabfall des Codes hin.

Denn, obwohl diese Tools in der Lage sind, schnell funktionalen Code zu produzieren, warnen Studien davor, dass den Ergebnissen oft die Expertise und Optimierung erfahrener Entwickler*innen fehlen (Pereira, et al., 2025). Nutzer*innen sollten sich bewusst sein, dass all dies langfristig insgesamt zum Verlust des Gesamtverständnisses (Butler, Suh, Haniyur, & Hadley, 2025), einem erhöhten Wartungsaufwand und einer Zunahme technischer Schulden führen kann (Pereira, et al., 2025).

3.3.10 Kompetenzabfall & Abhängigkeit

Ein weiteres Risiko, das in der Literatur im Zusammenhang mit der Nutzung generativer KI thematisiert wird, ist ein möglicher Kompetenzabfall (Deskilling) sowie die Entwicklung einer Abhängigkeit von den angewandten Tools. Entwickler*innen äußerten die Sorge, dass eine übermäßige Abhängigkeit dazu führen könnte, dass essenzielle Problemlösekompetenzen verloren gehen. Ebenso fällt auf, dass diese Gefahr vor allem bei weniger erfahrenen Entwickler*innen als kritisch eingestuft wird (Pereira, et al., 2025).

Lernende haben oft Schwierigkeiten, den von der KI generierten Gesamtkontext zu verstehen, was dazu führt, dass sie den Code übernehmen, ohne dessen Bedeutung komplett zu verstehen. Dies könnte dazu führen, dass Junior-Entwickler wichtige Chancen zum Lernen verpassen (Pereira, et al., 2025).

3.3.11 Automation Bias & Overreliance

Ein damit verbundenes menschliches Risiko ist der Automatisierungsbias (Automation Bias), bei dem Entwickler*innen den KI-Vorschlägen aufgrund ihrer oberflächlichen Korrektheit blind vertrauen. Diese Form des Übervertrauens (Overreliance) führt dazu, dass fehlerhafte Lösungen ohne gründliche Prüfung akzeptiert werden, was später wieder zu zeitaufwendigem Debugging führt (Weber, Brandmaier, Schmidt, & Mayer, 2024).

Besonders problematisch sind hierbei sogenannte Auslassungsfehler (Errors of Omission): Da KI-Tools fehlende Logik oder fehlende Informationen nicht identifizieren können, werden diese Lücken von den Anwender*innen möglicherweise übersehen. Fehlende Warnungen werden von vielen Entwickler*innen fälschlicherweise als Signal für Korrektheit des Codes interpretiert (Vasconcelos, Bansal, Fourney, Liao, & Vaughan, 2024).

3.3.12 Validierungsaufwand

All dies führt schließlich zu einem nicht unbedeutenden, zusätzlichen Validierungsaufwand, der den ursprünglichen Zeitvorteil und die mentale Entlastung, (Butler, Suh, Haniyur, & Hadley, 2025) teilweise wieder zunichtemachen kann. In den Ergebnissen der Studie von Urek et al. (2025), speziell in einer Umfrage unter professionellen Softwareentwickler*innen, wurde ermittelt, dass 22,4 % der Teilnehmer den generierten Code als komplett unbrauchbar ansahen. 45,9 % von ihnen berichteten, dass viel Nacharbeit in Form von Fehlerbehebungen erforderlich ist, bevor der generierte Code benutzt werden kann. Da KI-Modelle dazu neigen, Code zu produzieren, der zwar oberflächlich korrekt erscheint, aber trotzdem Fehler enthält (Butler, Suh, Haniyur, & Hadley, 2025), ist das Debugging oft zeitaufwendiger als bei selbst geschriebenem Code.

3.3.13 Datenschutz & Ethische Bedenken

Die Integration von GenAI-Werkzeugen in die Softwareentwicklung wird zudem von Bedenken hinsichtlich des Datenschutzes sowie ethischer und rechtlicher Fragen begleitet.

Studien belegen ein vorhandenes Misstrauen gegenüber der Sicherheit von KI-Systemen. Eine Mehrheit der befragten Entwickler*innen (72,9 %) einer Studie entfernt bereits sensible Informationen wie Lizenzen, kryptografische Schlüssel oder vertrauliche Unternehmensdaten manuell, bevor sie Code-Fragmente in KI-Tools übertragen. Über die Hälfte äußerte zudem eine hohe Besorgnis darüber, dass ihre Eingaben ungewollt gespeichert werden könnten (Urek, Maslać, & Budin, 2025).

Auch die Herkunft der Trainingsdaten spielt für viele Nutzer*innen eine wichtige Rolle. In einer Analyse von Marktplatz-Reviews bilden ethische Bedenken eine der am häufigsten geäußerten negativen Einflüsse auf die Erfahrung bei der Arbeit mit generativer KI (Lyu, et al., 2025). Nutzer kritisieren dabei beispielsweise den Trend, dass KI-Modelle auf Basis von Open-Source-Code trainiert wurden, für deren Nutzung nun jedoch Gebühren erhoben werden.

Nicht zuletzt herrscht Unsicherheit darüber, ob KI unbeabsichtigt Code generieren könnte, der gegen bestehende Urheberrechte verstößt (Pereira, et al., 2025).

4 Beantwortung der Forschungsfragen und Diskussion

In diesem Kapitel werden zunächst die zentralen Forschungsfragen auf Basis der Syntheseergebnisse zusammenfassend beantwortet, bevor die Ergebnisse im Kontext der Softwareentwicklung diskutiert und interpretiert werden.

4.1 Beantwortung RQ1

Die Analyse der 13 Primärstudien zeigt, dass generative KI in der Implementierungsphase weit über die reine Codeerzeugung hinaus als multifunktionales Assistenzsystem agiert. Ihre Rolle lässt sich grob in vier zentrale Funktionsbereiche unterteilen.

Als am weitesten verbreiteter Einsatzbereich wurde in den untersuchten Primärstudien die Aufgabe als Code-Generator und -Editor identifiziert. Die Erstellung und Vervollständigung von Code aus natürlichen Sprachaufforderungen ist die am häufigsten dokumentierte Kernfunktion. Da etwa 81 % der Aktivitäten in realen Projekten auf der Modifikation bestehenden Codes basieren (Wang, Ning, Zhang, Liu, & Wang, 2024), liegt ein wesentlicher Fokus auf dem Refactoring und der iterativen Code-Optimierung.

Dazu kommt die Rolle als Analyst und Debugger. Die KI dient als interaktives Werkzeug zur Fehleranalyse, das Fehlermeldungen interpretiert und Korrekturvorschläge generiert. Bei der Behebung einfacher Bugs kann dabei eine Zeitoptimierung von bis zu 50 % erreicht werden (Preethi, Ragavan, Abinandhana, Umamaheswari, & Suvethaa, 2024).

Über diese technischen Kernaufgaben hinaus fungiert die Technologie zunehmend als interaktive Lern- und Wissensquelle, die unter anderem als moderner Ersatz für klassische Websuchen oder Fachportale beschrieben wird. Entwickler*innen nutzen sie als „virtuellen Kollegen“ zum Brainstorming von Problemlösungen oder zum schnellen Erlernen neuer Syntax und komplexer Logik.

Schließlich dient GenAI der kognitiven Entlastung, indem repetitive Routineaufgaben automatisiert werden. Diese mentale Entlastung ermöglicht es den Anwender*innen, ihren Fokus verstärkt auf anspruchsvollere Probleme zu richten.

4.2 Beantwortung RQ2

Die Integration von GenAI in den Entwicklungsprozess ist geprägt von einem Spannungsfeld zwischen Effizienzgewinnen und qualitativen Risiken. Die Ergebnisse lassen sich wie folgt zusammenfassen:

In Bezug auf den messbaren Nutzen dokumentiert die Literatur eine deutliche Diskrepanz zwischen idealisierten Laborbedingungen und der industriellen Praxis: Während kontrollierte Studien Produktivitätssteigerungen von bis zu 65 % verzeichnen (Weber, Brandmaier, Schmidt, & Mayer, 2024), liegen die realen Werte in Unternehmen moderater zwischen 21 % und 26 % Zeitersparnis (Paradis, et al., 2025). Dieser objektive Nutzen geht mit einer hohen subjektiven Akzeptanz einher, da viele Entwickler*innen von einem gesteigerten Produktivitätsgefühl und einer höheren Arbeitsfreude berichten, die vor allem aus der kognitiven Entlastung bei mühsamen Routineaufgaben resultiert. Besonders für weniger erfahrene Kräfte fungieren die Tools zudem als starke Lernunterstützung. Sie geben an, neue Programmiersprachen oder Konzepte durch die KI-Vorschläge schneller zu erfassen.

Demgegenüber stehen technische Risiken, die insbesondere die Qualität der Software betreffen. Besonders kritisch ist die Unzuverlässigkeit im Bereich der Software-Sicherheit, da empirische Analysen belegen, dass bis zu 30 % des generierten Codes Sicherheitslücken aufweisen (Fu, et al., 2025). Zudem stößt die Technologie bei steigender Komplexität an ihre Grenzen: Sobald Aufgaben über isolierte Algorithmen hinausgehen und projektweite Abhängigkeiten oder tiefes Kontextverständnis erfordern, sinkt die Qualität der generierten Antworten stark ab.

Auf der menschlichen Ebene führt die Integration von GenAI zu einem hohen Validierungs- und Korrekturaufwand. Dies bedeutet einen zusätzlichen mentalen Aufwand für die Überprüfung des KI-Outputs, was in der Praxis dazu führen kann, dass Entwickler*innen einen signifikanten Teil ihrer Zeit mit der Korrektur und Nachbearbeitung fehlerhafter Vorschläge verbringen. Damit einher geht die Gefahr eines übermäßigen Vertrauens in die oberflächlich korrekten Lösungen der KI, was dazu führt, dass Logikfehler oder Sicherheitsmängel bei fehlender gründlicher Prüfung übernommen werden. Langfristig identifiziert die Forschung zudem das Risiko eines Kompetenzabfalls. Insbesondere bei Junior-Entwickler*innen besteht die Sorge, dass durch die übermäßige Abhängigkeit von generierten Lösungen essenzielle Problemlösekompetenzen und das tiefe Verständnis für Code-Strukturen verloren gehen können.

Zusätzlich zu diesen technischen und menschlichen Faktoren treten schließlich Misstrauen gegenüber der Datensicherheit und ethische Bedenken auf. Diese ergeben sich primär aus der Nutzung von Open-Source-Code als Trainingsbasis sowie aus der rechtlichen Unsicherheit bezüglich potenzieller Urheberrechtsverletzungen und fehlender (oder falschen) Quellenangaben der generierten Inhalte.

4.3 Interpretation der Ergebnisse und Diskussion

Die Ergebnisse dieser Arbeit bestätigen die bestehenden Theorien von Urek, Maslać, & Budin (2025), dass generative KI die Softwareentwicklung nicht nur automatisiert, sondern den Beruf

der Softwareentwickler*innen transformiert. Die Rolle der KI entwickelt sich von einem reinen Werkzeug zu einem „aktiven Kollaborateur“, was darauf deutet, dass neue Kompetenzen im Software-Engineering erforderlich sein werden.

4.3.1 Effizienzgewinn vs. „Cognitive Tax“

Während Studien signifikante Produktivitätssteigerungen und Zeitersparnisse darlegen (Paradis, et al., 2025), wird dieser Vorteil oft durch den hohen Aufwand für die Prüfung des generierten Codes zunichte gemacht (Baralla, Ibba, & Tonelli, 2025). Entwickler*innen verbringen in der Praxis teilweise über 50 % ihrer Zeit mit der Validierung und Korrektur von KI-Vorschlägen (Lyu, et al., 2025).

Dies stellt eine neue Form der mentalen Belastung dar, die mit dem Begriff der kognitiven Schulden, auch "Cognitive Tax" genannt, beschrieben werden kann. Die Zeitersparnis bei Routineaufgaben wird durch den gleichbleibenden oder steigenden Aufwand bei beispielsweise der Fehlersuche in komplexen Logikstrukturen wieder ausgeglichen.

4.3.2 Verschiebung der Kernkompetenzen

Die Analyse zeigt, dass Entwickler*innen GenAI nicht nur für die reine Code-Erzeugung nutzen (Khojah, Mohamad, Leitner, & De Oliveira Neto, 2024). Die Integration von GenAI verschiebt also die Kernaufgaben von Entwickler*innen; weg von der reinen Programmierung hin zur Code-Review und zur Validierung (Weber, Brandmaier, Schmidt, & Mayer, 2024). Entwickler*innen entfernen sich dabei von einer hauptsächlich ausführenden Rolle der Codeerstellung hin zu einer eher überwachenden und steuernden Funktion im Entwicklungsprozess. Es bedarf daher beispielsweise der Fähigkeiten, die KI durch zielgerichtete Formulierung von Prompts zu steuern und die generierten Ergebnisse auf systematische Weise zu bewerten. Dies umfasst etwa die Prüfung hinsichtlich funktionaler Korrektheit, Sicherheit, Wartbarkeit. Hierbei lauern weiterhin die in der Synthese analysierten Gefahren wie der Automatisierungsbias, übermäßiges Vertrauen und Auslassungsfehler.

4.3.3 KI als Ergänzung der menschlichen Expertise

Es besteht ein Konsens in der Literatur, dass GenAI als Ergänzung zur menschlichen Expertise betrachtet werden sollte und nicht als deren Ersatz (Urek, Maslać, & Budin, 2025).

Viele Entwickler*innen sind sich bewusst, dass eine undurchdachte Nutzung gefährlich ist (Pereira, et al., 2025), da Modelle beispielsweise dazu neigen, unvollständige Anforderungen durch Raten (Halluzinationen) zu füllen, anstatt ergänzende Rückfragen zu stellen (Wu & Fard, 2025).

Um das Potenzial generativer KI gänzlich auszunutzen, müssen Entwickler*innen also ein tiefgehendes Verständnis für diese Werkzeuge besitzen (Preethi, Ragavan, Abinandhana, Umamaheswari, & Suvethaa, 2024). Die Qualität der Ergebnisse bleibt somit stark abhängig von der Fähigkeit des Menschen, die KI zu bedienen, präzise Prompts zu formulieren und schließlich die generierten Inhalte kritisch zu evaluieren.

4.3.4 Von isolierten Aufgaben zur Komplexität der realen Welt

Die Synthese der Ergebnisse verdeutlicht einen großen Unterschied zwischen der Leistungsfähigkeit generativer KI bei isolierten Problemen und Umgebungen und deren Anwendung in tatsächlichen industriellen Umgebungen. Während die untersuchten Tools bei einfachen algorithmischen Aufgaben wie dem *FizzBuzz*-Algorithmus äußerst positive Ergebnisse liefern (Urek, Maslać, & Budin, 2025), sinkt die Erfolgsrate bei steigender Komplexität ab (Wang, Ning, Zhang, Liu, & Wang, 2024).

Ein Kernproblem in der Anwendung in der realen Praxis ist hierbei das mangelnde Kontextbewusstsein der Modelle (Baralla, Ibba, & Tonelli, 2025). In großen Unternehmenssystemen scheitert die Lösung komplexer Probleme mithilfe von KI oft daran, dass projektübergreifende Abhängigkeiten in beispielsweise derzeit nicht geöffneten Dateien nicht berücksichtigt werden, da die KI keine Übersicht über das Gesamtprojekt besitzt (Khojah, Mohamad, Leitner, & De Oliveira Neto, 2024).

Nutzer berichten in diesem Zusammenhang, dass die zusätzliche Bereitstellung des benötigen Kontexts und die Erklärung von Details im Endeffekt sogar mehr Zeit beansprucht als die manuelle Implementierung (Urek, Maslać, & Budin, 2025).

KI-Tools beschleunigen weiterhin primär die mühsame Arbeit, wie die Umsetzung von Algorithmen und die fehlerfreie Anwendung der Syntax. Die eigentliche Softwareentwicklung, welche weitere Faktoren wie Design und Architekturen umfasst, bleibt hingegen weiterhin eine Domäne menschlicher Expertise (Wang, Ning, Zhang, Liu, & Wang, 2024).

4.3.5 Implikationen für Praxis

Die Einführung von KI-Tools muss zwingend von durchgängigen Sicherheitsanalysen begleitet werden. Studien zeigen, dass etwa 25 % bis 30 % des generierten Codes Sicherheitslücken enthalten können (Fu, et al., 2025), weshalb eine Überprüfung durch qualifizierte Entwickler*innen vor deren Integration in die Produktion unerlässlich ist.

Unternehmen sollten klare Richtlinien für den verantwortungsvollen Einsatz von KI schaffen, die den verantwortungsvollen Umgang mit KI-Systemen definieren, etwa im Hinblick auf Einsatzgrenzen, Qualitätssicherung und Datenschutz.

Ein weiterer wichtiger Faktor ist der gezielte Kompetenzaufbau der Entwickler*innen. Neben klassischen Softwareentwicklungs-Fähigkeiten gewinnen insbesondere Kompetenzen im Umgang mit generativen KI-Tools an Bedeutung, darunter effektives Prompting oder die kritische Evaluation von generierten Ergebnissen.

4.3.6 Implikationen für Forschung

Zukünftige Forschung sollte über die isolierte Betrachtung kurzfristiger Produktivitätsgewinne, etwa in Form von Zeitersparnissen bei einfachen Programmieraufgaben, hinausgehen. Wie Butler, Suh, Haniyur, & Hadley (2025) fordern, bedarf es hierfür langfristiger Beobachtungszeiträume von bis zu einem Jahr. Es ist außerdem notwendig, Metriken spezifisch für die Softwareentwicklung zu erarbeiten, die Aspekte wie z. B. die langfristige Wartbarkeit, die Code-Qualität und die Auswirkungen auf die Teamkommunikation berücksichtigen (Vasconcelos, Bansal, Fourney, Liao, & Vaughan, 2024).

Darüber hinaus ist eine stärkere Untersuchung der Mensch-KI-Interaktion im Softwareentwicklungsprozess sinnvoll. Insbesondere sollten Mechanismen erforscht werden, die die Zuverlässigkeit und Transparenz generativer KI-Systeme erhöhen. Dazu gehören beispielsweise die Integration von Unsicherheitssignalen, die es Entwickler*innen ermöglichen, die Vertrauenswürdigkeit generierter Antworten besser einzuschätzen (Vasconcelos, Bansal, Fourney, Liao, & Vaughan, 2024), sowie die Fähigkeit von KI-Systemen, kontextabhängig Rückfragen zu stellen und so Missverständnisse beim Prompting zu reduzieren (Wu & Fard, 2025).

4.4 Limitationen der eigenen Arbeit

Trotz der systematischen Vorgehensweise unterliegt die vorliegende Arbeit folgenden Limitationen, die bei der Interpretation der Ergebnisse berücksichtigt werden müssen.

4.4.1 Restriktionen durch das Forschungsdesign

Aufgrund des begrenzten Bearbeitungszeitraums konnte keine vollumfängliche systematische Literaturanalyse (vgl. Kapitel 2.1) durchgeführt werden. Stattdessen wurde eine sehr zielgerichtete Suchstrategie gewählt, um die Anzahl der zu analysierenden Studien auf ein im Rahmen dieser Arbeit bearbeitbares Maß zu begrenzen. Diese Entscheidung birgt angesichts der zahlreichen Publikationen in diesem Bereich das Risiko, relevante Erkenntnisse nicht berücksichtigt zu haben.

Die Restriktion ergibt sich konkret aus der Struktur des verwendeten Suchstrings (vgl. Kapitel 2.3.2), wodurch ausschließlich Studien erfasst wurden, die all diese Konzepte explizit erwähnten. Beispielsweise könnte etwa die Kombination der Begriffe „productivity“ und „code“ zu einer

Verzerrung der Ergebnisse geführt haben. Während „productivity“ beispielsweise gezielt auf ein Nutzenpotenzial (RQ2) eingeht, wurde kein Stichwort für ein mögliches Risiko im Suchstring inkludiert. Weiterhin könnten Studien existieren, die relevante Nutzungsformen generativer KI im Softwareentwicklungsprozess untersuchen, die nicht direkt mit Code in Verbindung stehen (wie etwa das Erlernen von Konzepten oder die Unterstützung beim Brainstorming); wenn sie das Schlagwort „code“ jedoch nicht verwenden, wurden sie potenziell ausgeschlossen.

4.4.2 Potenzieller Confirmation Bias

Da diese Literaturanalyse auf der Arbeit einer einzelnen Person beruht, kann nicht vollständig ausgeschlossen werden, dass die Analyse trotz eines systematischen Vorgehens einem gewissen Bestätigungsfehler (Confirmation Bias) unterliegt. Dieser beschreibt die Tendenz, Informationen bevorzugt so wahrzunehmen, auszuwählen und zu deuten, dass sie bestehende Annahmen oder Erwartungen stützen, anstatt ihnen kritisch entgegenzustehen.

4.4.3 Aktualität der Technologie

Da sich generative KI in einem schnell entwickelnden Umfeld befindet, stellen die hier aufgezeigten Ergebnisse nur eine Momentaufnahme dar. Es könnten bereits neue Technologien existieren, welche die identifizierten Limitationen (z. B. Rückfragekompetenz (Vasconcelos, Bansal, Fourney, Liao, & Vaughan, 2024)) teilweise überwunden haben.

4.5 Limitationen der analysierten Literatur

Die wissenschaftliche Literatur zum Einsatz von GenAI im Software-Engineering weist mehrere systematische Limitationen auf, welche die Validität der Ergebnisse einschränken.

4.5.1 Art der Aufgabenstellungen

Ein wesentliches Muster in der Literatur ist die Nutzung von kontrollierten Umgebungen und vereinfachten Programmieraufgaben. Einige Studien basieren auf kleinen Programmieraufgaben oder Algorithmen-Rätseln, wie dem *FizzBuzz*-Algorithmus (Urek, Maslać, & Budin, 2025) oder Aufgaben von Plattformen wie LeetCode[16] im einfachen Schwierigkeitsgrad (Vasconcelos, Bansal, Fourney, Liao, & Vaughan, 2024). Diese spiegeln oft nicht die Komplexität industrieller Großprojekte wieder, in denen das Verständnis großer Codebasen oder auch

[16] https://leetcode.com/ [abgerufen am 26.04.2026]

die Systemarchitektur von Bedeutung sein können. Die Übertragbarkeit ist somit eingeschränkt.

4.5.2 Evaluationsmethoden

Während einige Studien detaillierte Sicherheitsanalysen durchführen (Fu, et al., 2025), konzentrieren sich andere rein auf funktionale Korrektheit, Zeitersparnis oder Anzahl an abgeschlossenen Aufgaben und ließen den Aspekt der Codequalität außen vor (Paradis, et al., 2025), wodurch qualitative Risiken unterrepräsentiert sein können.

4.5.3 Subjektivität

Mehrere Studien befassten sich mit der subjektiven Wahrnehmung der Teilnehmer, da sie auf Selbstauskunft von Entwickler*innen in Tagebuchstudien (Butler, Suh, Haniyur, & Hadley, 2025) oder Umfragen (Urek, Maslać, & Budin, 2025) basieren.

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung der Ergebnisse

Die vorliegende Bachelorarbeit hat im Rahmen einer systematischen Literaturanalyse den aktuellen Forschungsstand zum Einsatz generativer KI in der Implementierungsphase der Softwareentwicklung untersucht. Ziel war es, anhand der Forschungsfragen die konkreten Einsatzzwecke, die dokumentierten Nutzenpotenziale sowie die damit verbundenen Risiken und potenzielle Herausforderungen strukturiert zusammenzuführen.

Die Auswertung der 13 Primärstudien zeigt, dass GenAI weit über die reine Codeerzeugung hinaus als multifunktionales Assistenzsystem fungiert, das Aufgaben wie das Refactoring, das Debugging und die Wissensbeschaffung unterstützt und zunehmend die klassische Websuche ersetzt. Während die Literatur konsistent von Produktivitätssteigerungen und einer kognitiven Entlastung bei Routineaufgaben berichtet, zeigt die Analyse auch, dass diese Vorteile durch signifikante qualitative Risiken – insbesondere im Bereich der Software-Sicherheit und der mangelnden Kontextkompetenz der Modelle – relativiert werden. Auf der menschlichen Ebene verschärfen diese qualitativen Defizite das Risiko eines Kompetenzabfalls (Deskilling), wodurch langfristig essenzielle Problemlösefähigkeiten geschwächt werden können.

Ein zentrales Ergebnis der Arbeit ist die Erkenntnis, dass generative KI die Natur der Entwicklungsarbeit grundlegend verändert. Die Rolle der Softwareentwickler*innen verschiebt sich weg von einer primär schreibenden Tätigkeit hin zu einer verstärkt prüfenden und validierenden Funktion. Diese neue Form der Mensch-KI-Kollaboration erfordert zusätzliche Kompetenzen in der kritischen Evaluation von KI-Vorschlägen, um Gefahren wie dem Automatisierungsbias oder einem langfristigen Kompetenzabfall entgegenzuwirken.

So markiert der Einzug generativer KI nicht das Ende der klassischen Programmierung, sondern den Beginn einer neuen Ära der Mensch-KI-Kollaboration, in der die technologische Effizienz erst durch die kritische Urteilskraft des Menschen ihren vollen, sicheren Wert entfaltet.

5.2 Ausblick

Trotz der rasanten Fortschritte steht die Integration generativer KI in die industrielle Praxis erst am Beginn ihrer Entwicklung (Paradis, et al., 2025), weshalb sich aus den vorliegenden Erkenntnissen wichtige Perspektiven für die technologische sowie methodische Weiterentwicklung ergeben.

5.2.1 Kommunikations- und Kontextkompetenz

Zukünftige Entwicklungen generativer KI sollten verstärkt auf die Verbesserung der Kommunikations- und Kontextkompetenz abzielen. Aktuelle Modelle generieren Antworten primär auf Basis statistischer Wahrscheinlichkeiten, ohne ein tiefgehendes Verständnis des projektweiten Kontexts zu besitzen. Dies kann zu mehrdeutigen, unvollständigen oder fehlerhaften Ergebnissen führen.

Ein vielversprechender Ansatz besteht daher in der Weiterentwicklung interaktiver Fähigkeiten, insbesondere der Fähigkeit, kontextabhängig klärende Rückfragen zu stellen, welche von Wu & Fard (2025) untersucht wurde. Langfristig könnte dies die Verlässlichkeit der generierten Antworten erhöhen. Ergänzend hierzu bietet die visuelle Kommunikation von Modellunsicherheiten durch das sogenannte „Uncertainty Highlighting“ ein erhebliches Potenzial. Systeme könnten durch das Hervorheben unsicherer Tokens die Aufmerksamkeit der Entwickler*innen gezielt auf potenzielle Fehlerquellen lenken (Vasconcelos, Bansal, Fourney, Liao, & Vaughan, 2024).

5.2.2 Adaptive Unterstützung und Kompetenzentwicklung

Einige Quellen betonen, dass der Nutzen von KI-Tools stark von individuellen Faktoren der Anwender*innen abhängt (Paradis, et al., 2025). Vor diesem Hintergrund erscheint die Entwicklung adaptiver Systeme vielversprechend, die ihre Unterstützung dynamisch an das jeweilige Kompetenzniveau der Entwickler*innen anpassen.

Denn während erfahrene Entwickler*innen generative KI häufig als Inspirationswerkzeug nutzen, besteht bei weniger erfahrenen Anwender*innen die Gefahr eines schleichenden Kompetenzverlusts durch übermäßige Abhängigkeit (Deskilling) (Pereira, et al., 2025). Zukünftige Systeme könnten daher, so wie von Preethi, Ragavan, Abinandhana, Umamaheswari, & Suvethaa (2024) dargelegt, gezielt Methoden integrieren, die nicht nur produktivitätssteigernd wirken, sondern gleichzeitig Lernprozesse unterstützen und ein reflektiertes Arbeiten fördern.

5.2.3 Integration in die akademische und betriebliche Ausbildung

Die Ergebnisse dieser Arbeit verdeutlichen, dass die Integration generativer KI über eine rein technische Neuerung hinausgeht und einen Paradigmenwechsel darstellt, der eine Neuausrichtung der akademischen und betrieblichen Ausbildung erfordert.

Ein zentraler Aspekt ist die Fokussierung auf Systemarchitektur, Design und kritisches Review. Zwar beherrschen LLMs die Erzeugung des Programmcodes (Syntax und isolierte Algorithmen), jedoch scheitern sie oft an der eigentlichen Entwicklungsarbeit (komplexe Architekturen

und projektweiter Kontext) (Wang, Ning, Zhang, Liu, & Wang, 2024). Die Ausbildung sollte daher weniger Gewicht auf das Auswendiglernen von Syntax legen und stattdessen die Fähigkeit fördern, eine kritische Urteilskraft zu entwickeln, um komplexe Probleme zu zerlegen und KI-Vorschläge auf funktionale Korrektheit, Wartbarkeit und Sicherheit zu prüfen.

Literaturverzeichnis

[S1] Baralla, G., Ibba, G., & Tonelli, R. (2025). Assessing GitHub Copilot in Solidity Development: Capabilities, Testing, and Bug Fixing. *IEEE Access, vol. 12*, 164389-164411. doi:10.1109/ACCESS.2024.3486365

[S2] Butler, J., Suh, J., Haniyur, S., & Hadley, C. (2025). Dear Diary: A Randomized Controlled Trial of Generative AI Coding Tools in the Workplace. *47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 319-329. doi:10.1109/ICSE-SEIP66354.2025.00034

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 833-859. doi:10.1016/j.infsof.2008.01.006

[S3] Fu, Y., Liang, P., Tahir, A., Li, Z., Shahin, M., Yu, J., & Chen, J. (2025). Security weaknesses of Copilot-Generated Code in GitHub Projects: an empirical study. *ACM Transactions on Software Engineering and Methodology*, 1-34. doi:10.1145/3716848

[S4] Khojah, R., Mohamad, M., Leitner, P., & De Oliveira Neto, F. G. (2024). Beyond Code Generation: An observational study of ChatGPT usage in software engineering practice. *Proceedings of the ACM on Software Engineering*, 1819-1840. doi:10.1145/3660788

Kitchenham, B. (2004). Procedures for Performing Systematic Reviews. *Keele, UK, Keele Univ.*

[S5] Lyu, Y., Yang, Z., Shi, J., Chang, J., Liu, Y., & Lo, D. (2025). "My productivity is boosted, but …" Demystifying Users' Perception on AI Coding Assistants. *IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 191-203. doi:10.1109/ASE63991.2025.00024

[S6] Paradis, E., Grey, K., Madison, Q., Nam, D., Macvean, A., & Meimand, V. (2025). How Much Does AI Impact Development Speed? an Enterprise-Based Randomized Controlled Trial. *IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 618-629. doi:10.1109/ICSE-SEIP66354.2025.00060

[S7] Pereira, G. V., Jackson, V., Prikladnicki, R., Hoek, A. v., Fortes, L., Araújo, C., . . . Ramos, D. (2025). Exploring GenAI in Software Development: Insights from a Case Study in a Large Brazilian Company. *IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 330-341. doi:10.1109/ICSE-SEIP66354.2025.00035

[S8] Preethi, P., Ragavan, V., Abinandhana, C., Umamaheswari, G., & Suvethaa, D. R. (2024). Towards CodeBlizz: Developing an AI-Driven IDE Plugin for Real-Time Code Suggestions, Debugging, and Learning Assistance with Generative AI and Machine Learning Models. *International Conference on Emerging Research in Computational Science (ICERCS)*, 1-7. doi:10.1109/ICERCS63125.2024.10895857

[S9] Urek, I., Maslać, K., & Budin, A. (2025). An Insight into the Impact of Generative Artificial Intelligence Tools on Software Development Practices. *MIPRO 48th ICT and Electronics Convention*, 1909-1924. doi:10.1109/MIPRO65660.2025.11131852

[S10] Vasconcelos, H., Bansal, G., Fourney, A., Liao, Q. V., & Vaughan, J. W. (2024). Generation probabilities are not enough: uncertainty highlighting in AI code completions. *ACM Transactions on Computer-Human Interaction*, 1-30. doi:10.1145/3702320

[S11] Wang, W., Ning, H., Zhang, G., Liu, L., & Wang, Y. (2024). Rocks Coding, Not Development: A Human-Centric, Experimental evaluation of LLM-Supported SE tasks. *Proceedings of the ACM on Software Engineering, 1(FSE)*, 699-721. doi:10.1145/3643758

[S12] Weber, T., Brandmaier, M., Schmidt, A., & Mayer, S. (2024). Significant Productivity Gains through Programming with Large Language Models. *Proceedings of the ACM on Human-Computer Interaction, 8(EICS)*, 1-29. doi:10.1145/3661145

[S13] Wu, J. J., & Fard, F. H. (2025). HumanEvalComm: Benchmarking the communication com-petence of code generation for LLMs and LLM Agent. *ACM Transactions on Software Engineering and Methodology*, 1-42. doi:10.1145/3715109

Nutzung von Künstlicher Intelligenz

Bei der Erstellung dieser Bachelorarbeit wurden verschiedene Werkzeuge auf Basis Künstlicher Intelligenz unterstützend eingesetzt. Die Nutzung erfolgte in folgenden Bereichen:

- **NotebookLM**[17]: Dieses Tool wurde primär zur Unterstützung des Textverständnisses komplexer Quellen genutzt. Zusätzlich diente es als Hilfsmittel für Formulierungsvorschläge sowie zur abschließenden Grammatik- und Rechtschreibprüfung der Arbeit.
- **ChatGPT**: Die Anwendung kam zu Beginn der Arbeit für das initiale Brainstorming und die Entwicklung einer logischen Gliederung der Inhalte der Arbeit zum Einsatz. Zudem wurde das Tool für das Umschreiben einzelner Textpassagen genutzt, um die Lesbarkeit oder Verständlichkeit zu erhöhen.
- **DeepL**[18]: Ergänzend wurde DeepL zur Unterstützung bei der Übersetzung des Abstracts in die englische Sprache sowie bei dem gezielten Umschreiben für die Verbesserung der sprachlichen Varianz eingesetzt.

[17] https://notebooklm.google.com [abgerufen am 27.04.2026]
[18] https://www.deepl.com [abgerufen am 27.04.2026]

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit mit dem Titel:

Einsatz von generativer KI in der Softwareentwicklung: Eine Systematische Literaturanalyse

selbständig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen habe, habe ich deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.

03.05.2026
Datum

[redacted]