KI in der Mensch-Computer-Interaktion: Auswirkungen von LLMs auf die Softwareentwicklung

Simon Hörtzsch
TU Bergakademie Freiberg
Freiberg, Germany
Simon.Hoertzsch@student.tu-freiberg.de

Abstract

In Zeiten von leistungsstarken "Large Language Models" (LLMs) wie ChatGPT und Gemini wird deren Einfluss auf den gesamten Softwareentwicklungsprozess untersucht. Diese Seminararbeit analysiert, wie KI-Technologien nicht nur die reine Programmierung, sondern den kompletten Lebenszyklus der Softwareentwicklung – von der Anforderungsanalyse über die Erstellung von User Personas und dem UI/UX-Design bis hin zum Testen – verändern [16]. Es wird der Frage nachgegangen, inwieweit LLMs als kollaborative Partner agieren können [14, 18], um die Effizienz zu steigern und kreative Prozesse zu unterstützen. Dabei werden sowohl die enormen Potenziale zur Automatisierung und Unterstützung im Design- und Forschungsprozess beleuchtet [16, 18] als auch kritische Herausforderungen wie das "Black-Box"-Problem [17], die Notwendigkeit von Erklärbarkeit und die signifikanten Sicherheitsrisiken in KI-generiertem Code [10] diskutiert. Ziel ist es, eine differenzierte Betrachtung der Chancen und Risiken zu ermöglichen und die Notwendigkeit eines Human-in-the-Loop-Ansatzes zu untermauern [14, 17], um eine verantwortungsvolle und effektive Integration von LLMs in die Softwareentwicklung zu gewährleisten.

Keywords

Mensch-Computer-Interaktion, KI, LLM, Effizienz, Softwareentwicklung, UI/UX-Design, Black-Box-Problem

ACM Reference Format:
Simon Hörtzsch. 2025. KI in der Mensch-Computer-Interaktion: Auswirkungen von LLMs auf die Softwareentwicklung. In *Proceedings of Seminar on Ubiquitous and Interactive Systems (UbiSys Seminar '25).* ACM, New York, NY, USA, 7 pages.

1 Einleitung

Die rasante Entwicklung von "Large Language Models" (LLMs) hat begonnen, traditionelle Arbeitsweisen in vielen Branchen grundlegend zu verändern. Insbesondere im Bereich der Softwareentwicklung entfalten diese Technologien ein transformatives Potenzial, das weit über die reine Codegenerierung hinausgeht. Während frühe Diskussionen sich oft auf die Fähigkeit von KI-Assistenten zur Erstellung von Code-Snippets konzentrierten, wird heute deutlich, dass ihr Einfluss den gesamten Softwareentwicklungszyklus umfasst: von der initialen Anforderungsanalyse und der Erstellung von Systemarchitekturen über das Design von User Interfaces (UI) und die Generierung von User Personas bis hin zu automatisierten Testverfahren und der Wartung von Systemen [16].

Diese Entwicklungen werfen zentrale Fragen im Forschungsfeld der Mensch-Computer-Interaktion (HCI) auf. Inwieweit können LLMs als intelligente Werkzeuge oder sogar als kollaborative Partner fungieren, um Entwicklerteams zu unterstützen und die Effizienz zu steigern [14, 16]? Welchen Einfluss hat die Integration von KI auf kreative Prozesse wie das UX-Design, und wie verändert sich die Rolle des menschlichen Entwicklers in diesem neuen Paradigma der "Mensch-Computer-Co-Kreativität" [9]?

Um die Leistungsfähigkeit von LLMs im Kontext der Softwareentwicklung zu bewerten, werden spezialisierte Benchmarks eingesetzt. Während allgemeine Wissenstests primär grundlegende Fähigkeiten prüfen – wie etwa das breite Allgemeinwissen in MMLU (Massive Multitask Language Understanding) [6, 19], das Expertenwissen in GPQA (Graduate-Level Google-Proof Q&A) [11] oder das logische Denkvermögen in AIME (American Invitational Mathematics Examination) [2], treten für die Softwareentwicklung anwendungsorientierte Bewertungen in den Vordergrund. Spezialisierte Benchmarks wie LiveCodeBench [7] oder der SWE-bench [1] testen die Fähigkeit von Modellen, reale Probleme aus GitHub-Repositories zu lösen und bewerten somit direkt ihre Praxistauglichkeit im Software Engineering (SWE). Solche praxisnahen Tests sind entscheidend, um das wahre Potenzial und die Grenzen der aktuellen KI-Modelle einschätzen zu können [7].

Diese Seminararbeit hat zum Ziel, die vielfältigen Auswirkungen von LLMs auf den modernen Softwareentwicklungsprozess zu analysieren. Dabei wird ein besonderer Fokus auf den Paradigmenwechsel von der reinen Programmierung hin zu einem ganzheitlichen, KI-gestützten Entwicklungszyklus gelegt. Zunächst werden die grundlegenden Konzepte einer auf den Menschen zentrierten KI (Human-Centered AI) vorgestellt. Darauf aufbauend werden die Chancen durch KI in der HCI beleuchtet, insbesondere im Hinblick auf den Design- und Forschungsprozess. Anschließend erfolgt eine kritische Auseinandersetzung mit den Herausforderungen und Gefahren, wie dem "Black-Box"-Problem und Sicherheitsrisiken. Die Auswirkungen auf spezifische Anwendungsfelder, insbesondere das UI- und UX-Design, werden detailliert betrachtet, bevor die Arbeit mit einer umfassenden Diskussion und einem Fazit schließt.

UbiSys Seminar '25, TU Freiberg, DE

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM

2 Grundlagen

2.1 Künstliche Intelligenz im Kontext von HCI

Die Integration von Künstlicher Intelligenz (KI) in interaktive Systeme markiert einen fundamentalen Wandel in der HCI [17]. Traditionelle HCI-Paradigmen basierten oft auf einer "Reiz-Reaktions"-Beziehung, bei der ein System deterministisch auf explizite Nutzereingaben reagiert [17]. Moderne KI-Systeme hingegen funktionieren zunehmend autonom, lernen aus großen Datenmengen und können menschliches Verhalten antizipieren und darauf proaktiv reagieren. Diese Entwicklung verändert nicht nur die Art, wie wir mit Technologie interagieren, sondern auch den Prozess, wie diese Technologie entworfen und entwickelt wird.

Im Zentrum dieser Transformation steht das Konzept der Human-Centered AI (HCAI), das fordert, den Menschen in den Mittelpunkt der Entwicklung von KI-Systemen zu stellen [17]. Anstatt KI als reines Werkzeug zur Automatisierung zu betrachten, betont der HCAI-Ansatz die Notwendigkeit, KI-Systeme so zu gestalten, dass sie menschliche Fähigkeiten erweitern, anstatt sie zu ersetzen [17]. Dieser Ansatz zielt darauf ab, zuverlässige, sichere und vertrauenswürdige KI-Systeme zu schaffen [17]. Um dies zu erreichen, sind zwei zentrale Prinzipien von entscheidender Bedeutung: Human-in-the-Loop und Meaningful Human Control.

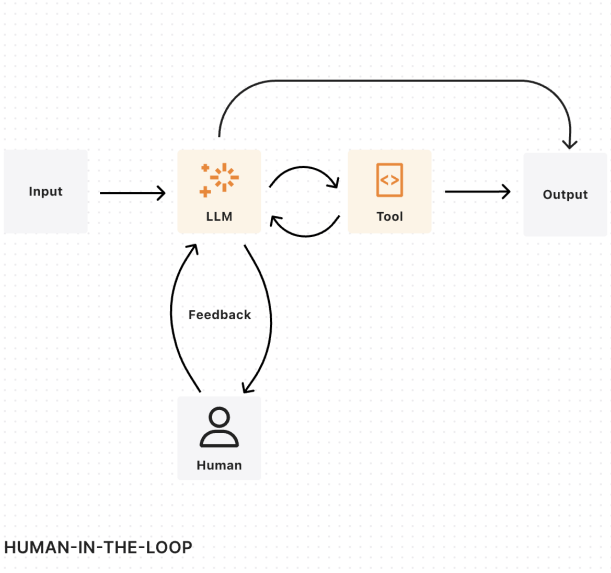

Figure 1: Das Prinzip "Human-in-the-Loop", bei dem menschliche Expertise gezielt in automatisierte KI-Workflows integriert wird. (Bildquelle: [4])

Human-in-the-Loop (Mensch im Kreislauf) beschreibt Systeme, in denen Menschen aktiv in den Lebenszyklus des KI-Modells eingebunden sind [14, 17] (siehe Abbildung 1). Im Kontext der Softwareentwicklung bedeutet dies, dass Entwickler nicht nur passive Konsumenten von KI-generiertem Code oder Designvorschlägen sind. Stattdessen sind sie aktive Teilnehmer, die das System trainieren, dessen Ergebnisse überprüfen und durch kontinuierliches Feedback verfeinern. Dies kann beispielsweise durch die Korrektur von KI-generiertem Code oder die Auswahl der besten aus mehreren Designvarianten geschehen, wodurch das Modell iterativ verbessert wird.

Meaningful Human Control (Sinnvolle menschliche Kontrolle) geht noch einen Schritt weiter und fordert, dass Menschen zu jeder Zeit die ultimative Autorität und Verantwortung über ein KI-System behalten [17]. Damit diese Kontrolle "sinnvoll" ist, müssen drei Bedingungen erfüllt sein [17]:

- Der Mensch muss die Handlungen des KI-Systems verstehen können. Dies adressiert das "Black-Box"-Problem vieler moderner KI-Modelle, bei denen die Entscheidungsprozesse intransparent sind [17].
- Der Mensch muss in der Lage sein, das System zu überwachen und dessen Verhalten nachzuvollziehen. Dies umfasst sowohl die Überprüfung der Eingaben als auch der Ausgaben des Systems.
- Der Mensch muss das System übersteuern können, um dessen Handlungen zu lenken oder zu korrigieren, insbesondere in kritischen Situationen.

Für die Softwareentwicklung bedeutet dies, dass Entwickler die Vorschläge einer KI nicht blind übernehmen dürfen. Sie müssen die Fähigkeit und die Werkzeuge besitzen, die Funktionsweise, die Grenzen und die potenziellen Schwachstellen des generierten Codes oder Designs zu verstehen. Nur so können sie die Verantwortung für die Qualität und Sicherheit des Endprodukts tragen.

Die Anwendung dieser Prinzipien ist entscheidend, um die Potenziale der KI in der Softwareentwicklung voll auszuschöpfen und gleichzeitig Risiken wie den Verlust von Kontrolle, die Einführung von Sicherheitslücken oder die Erzeugung von qualitativ minderwertigen Ergebnissen zu minimieren [17].

2.2 Aktuelle Benchmarks leistungsstarker LLMs

Um die Fähigkeiten von "Large Language Models" (LLMs) objektiv zu bewerten und ihre Eignung für die vielfältigen Aufgaben der Softwareentwicklung zu vergleichen, werden standardisierte Tests, sogenannte Benchmarks, eingesetzt. Eine ganzheitliche Bewertung erfordert dabei die Betrachtung verschiedener Fähigkeitsdimensionen. Während allgemeine Benchmarks primär grundlegende Fähigkeiten prüfen – wie etwa das breite Allgemeinwissen in **MMLU** [6], das Expertenwissen in **GPQA** [11] oder das logische Denkvermögen in **AIME** [2], sind sie nur bedingt aussagekräftig für die direkten, praxisnahen Anforderungen im Software Engineering [7]. Eine hohe Punktzahl in diesen Tests ist kein Garant für die Erzeugung von qualitativ hochwertigem und sicherem Code.

2.2.1 Spezialisierte Benchmarks für die Softwareentwicklung. Aus diesem Grund wurden spezialisierte Benchmarks entwickelt, die die Leistung von LLMs direkt im Kontext von Programmier- und Software-Engineering-Aufgaben messen. Die beiden in der nachfolgenden Tabelle verglichenen Ansätze sind:

- **SWE-bench:** Dieser Benchmark gilt als einer der realistischsten Tests zur Messung der Fähigkeit von LLMs, komplexe, reale Probleme aus bekannten GitHub-Repositories wie ‘django‘ oder ‘matplotlib‘ zu lösen [8]. Anstatt isolierter

Programmieraufgaben muss hier der gesamte Problemlösungsprozess innerhalb einer bestehenden Codebasis abgebildet werden.

- **LiveCodeBench:** Dieser Benchmark fokussiert sich auf die interaktive Natur des Programmierens, wie sie in Programmierwettbewerben vorkommt. Er bewertet die Fähigkeit eines Modells, in einer "Live-Coding"-Umgebung neue, zuvor ungesehene Probleme zu lösen, was ihn besonders robust gegenüber Datenkontamination macht [7].

2.2.2 Benchmark-Vergleich in der Praxis. Tabelle 1 zeigt einen direkten Leistungsvergleich aktueller KI-Modelle über diese verschiedenen Fähigkeitsdimensionen hinweg. Die Ergebnisse verdeutlichen, dass das Leistungsprofil der Modelle stark variiert. So kann ein Modell in der Logik (AIME) führend sein, aber bei der praktischen Fehlerbehebung in echten Repositories (SWE-bench) schwächer abschneiden. Besonders aufschlussreich ist der Vergleich zwischen SWE-bench und LiveCodeBench, da er die unterschiedlichen Stärken der Modelle bei der Arbeit mit bestehendem Code gegenüber neuen Programmieraufgaben aufzeigt. Ein weiterer wichtiger Faktor ist das **Kontextfenster**, welches die Fähigkeit eines Modells beeinflusst, komplexe Codebasen zu analysieren.

2.2.3 Kritische Betrachtung und Limitationen von Benchmarks. Obwohl spezialisierte Benchmarks eine deutliche Verbesserung darstellen, weisen sie weiterhin signifikante Limitationen auf. Sie messen oft nur die funktionale Korrektheit, lassen aber entscheidende qualitative Aspekte außer Acht, die in der professionellen Softwareentwicklung unerlässlich sind:

- **Sicherheit von KI-Output:** Die Sicherheit des generierten Codes wird in der Regel nicht geprüft. Eine umfassende Nutzerstudie von Perry et al. (2023) belegt dies eindrücklich: Entwickler, die Zugang zu einem KI-Assistenten hatten, schrieben signifikant häufiger unsicheren Code als die Kontrollgruppe [10]. Zudem waren sie eher davon überzeugt, sicheren Code verfasst zu haben, was auf eine gefährliche Überschätzung der eigenen Leistung hindeutet [10].
- **Wartbarkeit und Code-Qualität:** Benchmarks bewerten selten die Qualität des generierten Codes. Aspekte wie Lesbarkeit, Einhaltung von Design-Prinzipien (z.B. DRY - Don't Repeat Yourself), Modularität und Kommentierung werden nicht erfasst [7]. Ein LLM kann eine Aufgabe zwar "lösen", der erzeugte Code kann aber so komplex und schlecht strukturiert sein, dass er für menschliche Entwickler kaum wartbar ist.
- **Interaktive und kontextuelle Leistung:** Die meisten Benchmarks sind statisch und basieren auf einem einmaligen "Prompt-Antwort"-Schema. Sie bilden nicht den iterativen Dialog ab, der für die Softwareentwicklung typisch ist [7]. Ein Entwickler verfeinert seine Anfragen, bittet um Alternativen und baut auf vorherigen Ergebnissen auf. Diese kollaborative und kontextsensitive Interaktion wird von aktuellen Benchmarks kaum erfasst.
- **Ethische Implikationen und Bias:** KI-Modelle werden mit riesigen Mengen an Code aus öffentlichen Repositories trainiert. Diese Daten können veraltete Praktiken, Vorurteile oder ineffiziente Lösungsansätze enthalten. Benchmarks messen nicht, ob ein Modell diese negativen Muster reproduziert oder ob der generierte Code ethischen und inklusiven Standards entspricht [9].

Zusammenfassend lässt sich festhalten, dass Benchmarks zwar ein nützliches Werkzeug zur Leistungsmessung sind, ihre Ergebnisse jedoch kritisch hinterfragt werden müssen. Für eine ganzheitliche Bewertung der Eignung von LLMs in der Softwareentwicklung müssen qualitative Faktoren wie Sicherheit, Wartbarkeit und die Qualität der Mensch-KI-Kollaboration stärker in den Fokus rücken.

3 Chancen durch KI in der Softwareentwicklung

Die Integration von Künstlicher Intelligenz, insbesondere von generativen Modellen, eröffnet weitreichende Möglichkeiten, den kompletten Softwareentwicklungsprozess effizienter, effektiver und kreativer zu gestalten. Die Potenziale gehen dabei weit über die reine Code-Automatisierung hinaus und betreffen grundlegende Aspekte der Anforderungsanalyse, des Designs und der HCI [16]. KI agiert hierbei nicht nur als Werkzeug, sondern zunehmend als Assistenz- und Kollaborationspartner [14, 18].

3.1 Automatisierung und Unterstützung im Design- und Forschungsprozess

Eine der größten Chancen liegt in der Beschleunigung und qualitativen Verbesserung der frühen Phasen der Softwareentwicklung, die traditionell mit hohem manuellem Aufwand verbunden sind.

3.1.1 User Research und Analyse. Im Bereich der Nutzerforschung ermöglicht KI eine tiefgreifendere und datengestützte Analyse des Nutzerverhaltens. Anstatt sich ausschließlich auf subjektive Interviews zu verlassen, können Entwicklerteams objektive Analysedaten nutzen, um sogenannte "digitale Personas" zu erstellen [16]. Diese datengesteuerten Archetypen basieren auf echten Verhaltensmustern und erhöhen die Validität und Relevanz der Personas erheblich [16]. Zudem können Technologien wie Natural Language Processing (NLP) genutzt werden, um große Mengen an qualitativem Feedback, wie Interview-Transkripte oder Nutzerrezensionen, schnell und systematisch auszuwerten und darin wiederkehrende Muster oder Probleme (Pain Points) zu identifizieren [18].

3.1.2 UI/UX Design und Prototyping. Im UI- und UX-Design fungiert KI als Katalysator für Kreativität und Effizienz. Moderne Modelle sind in der Lage, aus einfachen, handgezeichneten Skizzen direkt funktionale UI-Wireframes oder sogar Code zu generieren [16]. Dieser Ansatz verkürzt den Weg von der Idee zum Prototyp drastisch. Darüber hinaus können generative KI-Systeme genutzt werden, um in kürzester Zeit hunderte von Designvarianten für A/B-Tests zu erstellen. Dies ermöglicht es Teams, fundierte, datenbasierte Entscheidungen über das Interface-Design zu treffen [16]. KI-Werkzeuge können zudem automatisiert überprüfen, ob Designentwürfe etablierten Design-Richtlinien und Usability-Heuristiken entsprechen [16].

Table 1: Vergleich relevanter KI-Modelle anhand ausgewählter Benchmarks. Die Daten spiegeln den Stand vom 25. August 2025 wider und stammen von Artificial Analysis [3], SWE-bench [12] und weiteren Quellen [5, 13].

Modell	MMLU (Allgemeinwissen)	GPQA (Expertenwissen)	AIME 2025 (Mathe/Logik)	SWE-bench (SWE, Coding)	LiveCodeBench (Coding)	Kontextfenster (in Tokens)
GPT-5	87,1%	85,4%	94,3%	65,0%	66,8%	400k
o3	85,3%	82,7%	88,3%	58,4%	78,4%	200k
Gemini 2.5 Pro	86,2%	84,4%	88,7%	53,6%	80,1%	~1M
Grok 4	86,6%	87,7%	92,7%	58,6%	81,9%	256k
DeepSeek R1	84,9%	81,3%	89,7%	57,6%	77,0%	128k
Claude 4 Opus Thinking	87,3%	79,6%	73,3%	67,6%	63,6%	200k

3.2 Generative KI als Co-kreativer Partner

Die vielleicht tiefgreifendste Veränderung betrifft die Rolle der KI im kreativen Prozess. Anstatt nur als Werkzeug zur Ausführung von Befehlen zu dienen, entwickelt sich die KI zu einem co-kreativen Partner. Dieser Paradigmenwechsel führt zu einer "Mensch-Computer-Co-Kreativität", bei der sowohl der Mensch als auch die KI kreative Vorschläge in den Entwicklungsprozess einbringen [9]. In diesem Modell übernimmt die KI nicht mehr nur die Rolle eines Problemlösers, sondern unterstützt auch bei der Problemfindung ("problem finding"), indem sie neue, unerwartete Perspektiven und Ideen generiert [9]. Für den Softwareentwickler bedeutet dies eine Verlagerung des Fokus: Anstatt sich auf die Implementierung von Details zu konzentrieren, kann er sich stärker auf übergeordnete, architektonische und kreative Entscheidungen fokussieren, während die KI bei der Ausarbeitung unterstützt.

3.3 Personalisierung und Intelligente User Interfaces (IUI)

KI-Technologien sind der Schlüssel zur Entwicklung von Intelligenten User Interfaces (IUIs), die eine neue Stufe der Personalisierung und kontextuellen Anpassung ermöglichen. IUIs sind in der Lage, sich an den einzelnen Nutzer, seinen aktuellen Kontext und seine Absichten anzupassen [16]. Im Gegensatz zu statischen Interfaces, die für einen "Durchschnittsnutzer" entworfen werden, können IUIs ihr Verhalten und ihre Darstellung dynamisch ändern. Multimodale KI-Modelle, die Text, Bild und Ton verarbeiten, eröffnen hierbei völlig neue Möglichkeiten zur Individualisierung.

Ein konkretes Beispiel aus der Softwareentwicklung: Ein Programmierer arbeitet an einem spezifischen Problem in einer komplexen Codebasis. Ein System mit IUI könnte dies erkennen und automatisch relevante Abschnitte aus der internen Dokumentation, passende Code-Beispiele aus früheren Projekten oder sogar Warnungen vor potenziellen Seiteneffekten seiner aktuellen Änderungen anzeigen [16]. Dies steigert nicht nur die Effizienz, sondern kann auch die Code-Qualität und das Verständnis für das Gesamtsystem nachhaltig verbessern.

4 Herausforderungen und Gefahren

Trotz der vielfältigen Chancen birgt die Integration von KI in die Softwareentwicklung auch erhebliche Herausforderungen und Risiken, die ein tiefes Verständnis und einen bewussten Umgang erfordern. Die bloße Anwendung von KI als Werkzeug zur Effizienzsteigerung reicht nicht aus; es ist entscheidend, die inhärenten Probleme der Technologie zu adressieren, um negative Konsequenzen zu vermeiden [17].

4.1 Das "Black-Box"-Problem und die Notwendigkeit von Erklärbarkeit

Eine der fundamentalsten Herausforderungen aktueller KI-Modelle, insbesondere von "Deep Learning"-Systemen, ist ihre Intransparenz [16]. Oftmals ist es selbst für Experten nicht nachvollziehbar, wie ein Modell zu einer bestimmten Ausgabe gelangt. Dieser Mangel an Nachvollziehbarkeit wird als das **"Black-Box"-Problem** bezeichnet [17] (siehe Abbildung 2).

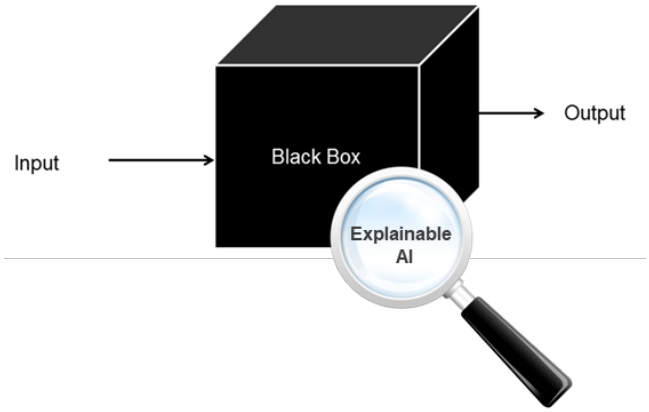

Figure 2: Visualisierung des Black-Box-Problems in der KI, bei dem die internen Prozesse intransparent sind. Explainable AI (XAI) versucht, diese Box aufzubrechen. (Bildquelle: [15])

Für die Softwareentwicklung hat dies gravierende Folgen:

- **Untergrabenes Vertrauen und komplizierte Fehlersuche:** Wenn ein KI-Assistent bestimmten Code oder spezielle Designs vorschlägt, der Entwickler aber nicht verstehen kann, warum genau weshalb, dann untergräbt dies das Vertrauen in das Werkzeug. Im Fehlerfall wird die Fehlersuche (das Debugging) erheblich erschwert, da die zugrundeliegende Logik des generierten Artefakts unklar ist [17].
- **Unmögliche Risikobewertung:** Ohne Transparenz ist eine fundierte Risikobewertung des von der KI generierten

Codes oder Designs unmöglich. Entwickler können nicht sicher beurteilen, ob alle Randfälle bedacht oder potenzielle Schwachstellen vermieden wurden.

Als Lösungsansatz hat sich das Forschungsfeld der **Explainable AI (XAI)** etabliert. Ziel von XAI ist es, die Entscheidungen von KI-Modellen transparent, interpretierbar und für den Menschen nachvollziehbar zu machen [17]. Anstatt nur ein Ergebnis zu liefern, sollen XAI-Systeme auch eine verständliche Begründung für dieses Ergebnis bereitstellen. Dies ist eine entscheidende Voraussetzung, um das Prinzip der "Meaningful Human Control" umzusetzen und Entwicklern die Möglichkeit zu geben, die Vorschläge der KI fundiert zu bewerten und die letztendliche Verantwortung zu übernehmen.

4.2 Sicherheitsrisiken in KI-generiertem Code

Neben der Intransparenz stellt die Sicherheit von KI-generiertem Code eine der größten Gefahren dar. KI-Assistenten werden auf riesigen Mengen an öffentlich verfügbarem Code trainiert, der zwangsläufig auch unsichere Programmierpraktiken, veraltete Bibliotheksverwendungen und unentdeckte Schwachstellen enthält [10]. Die Modelle lernen diese Muster und können sie in ihren Vorschlägen reproduzieren.

Eine umfassende Nutzerstudie von **Perry et al. (2023)** untersuchte die Auswirkungen von KI-Assistenten auf die Sicherheit des von Entwicklern geschriebenen Codes. Die Ergebnisse sind alarmierend:

- Entwickler mit Zugang zu einem KI-Assistenten schrieben **signifikant häufiger Code mit kritischen Sicherheitslücken** als die Kontrollgruppe ohne KI-Unterstützung [10]. Dies zeigte sich über verschiedene Aufgaben hinweg, von der Kryptografie bis hin zur Abwehr von SQL-Injection-Angriffen.
- Teilnehmer, die die KI nutzten, waren zudem **eher davon überzeugt, sicheren Code geschrieben zu haben** [10]. Dieses Phänomen der "KI-induzierten Selbstüberschätzung" ist besonders gefährlich, da es die kritische Überprüfung der generierten Ergebnisse verringert und Entwickler in einem falschen Gefühl der Sicherheit wiegt.

Diese Ergebnisse verdeutlichen, dass die funktionale Korrektheit, die von Benchmarks oft als einziges Kriterium gemessen wird, nicht ausreicht. Die unkritische Übernahme von KI-generiertem Code stellt ein ernsthaftes Sicherheitsrisiko für Softwareprojekte dar. Es unterstreicht die Notwendigkeit, dass Entwickler nicht nur die Fähigkeit zur Bedienung von KI-Tools erlernen, sondern vor allem ihre Kompetenz in der kritischen Verifikation und im sicheren Programmieren schärfen müssen.

5 Auswirkungen auf die Softwareentwicklung

Der Einfluss von "Large Language Models" (LLMs) auf die Softwareentwicklung ist tiefgreifend und transformativ. Er beschränkt sich nicht auf die reine Codegenerierung, sondern erfasst den gesamten Lebenszyklus einer Anwendung – von der ersten Idee bis zur finalen Wartung. KI-Systeme entwickeln sich von reinen Werkzeugen zu aktiven Partnern im Entwicklungsprozess, was zu einer Neudefinition von Rollen und Arbeitsabläufen führt [17]. Diese Entwicklung wird im Folgenden anhand der zentralen Phasen des Softwareentwicklungsprozesses beleuchtet.

5.1 Anforderungsanalyse und Systemarchitektur

In der initialen Phase der Anforderungsanalyse können LLMs eine wertvolle Rolle bei der Verarbeitung und Strukturierung von Informationen spielen. Sie sind in der Lage, große Mengen unstrukturierter Daten – wie Kundenfeedback, Anforderungsdokumente oder Meeting-Transkripte – zu analysieren und daraus Kernaussagen, User Stories oder funktionale Anforderungen zu extrahieren [16]. Dies beschleunigt den Prozess der Anforderungserhebung erheblich und hilft, Inkonsistenzen oder fehlende Informationen frühzeitig zu erkennen.

Auch bei der Konzeption der Systemarchitektur können LLMs als "Sparringspartner" dienen. Entwickler können Architekturentwürfe in natürlicher Sprache beschreiben und die KI bitten, diese auf Basis etablierter Design-Patterns (z.B. Microservices, MVC) zu bewerten, potenzielle Schwachstellen aufzuzeigen oder alternative Lösungsansätze vorzuschlagen. Modelle mit großen Kontextfenstern sind hier besonders im Vorteil, da sie komplexe Abhängigkeiten innerhalb eines Systems besser nachvollziehen können.

5.2 UI/UX-Design und Prototyping

Im Bereich des User Interface (UI) und User Experience (UX) Designs entfaltet KI ihr volles Potenzial als kreativer und unterstützender Partner. Der traditionell aufwendige Prozess von der Idee zum Prototyp wird durch KI-gestützte Werkzeuge radikal beschleunigt und verbessert [16].

5.2.1 Datengesteuerte User Personas. Die Erstellung von User Personas, die traditionell auf Interviews und Umfragen basiert, wird durch KI datengesteuerter und objektiver. Anstatt manuell kleine Stichproben auszuwerten, können Algorithmen riesige Mengen an Nutzungsdaten analysieren, um Verhaltensmuster zu erkennen und daraus automatisch "digitale Personas" zu generieren [16]. Diese Personas repräsentieren reale Nutzergruppen mit einer höheren statistischen Validität und ermöglichen eine präzisere, zielgruppengerechte Gestaltung.

5.2.2 Automatisierung im Designprozess. Generative KI-Modelle können den Designprozess auf vielfältige Weise automatisieren und inspirieren:

- **Vom Sketch zum Code:** Moderne KI-Systeme können handgezeichnete Skizzen oder einfache Wireframes interpretieren und direkt in funktionsfähigen Code für Web- oder mobile Anwendungen umwandeln [16]. Dies verkürzt die Prototyping-Phase enorm.
- **Generierung von Designvarianten:** Für A/B-Tests können LLMs hunderte von Designalternativen für ein UI-Element oder eine ganze Seite erstellen [16]. So können Teams schnell und effizient testen, welche Variante bei den Nutzern am besten ankommt.
- **Einhaltung von Design-Richtlinien:** KI kann Entwürfe automatisiert daraufhin überprüfen, ob sie etablierten, gewünschten Design-Systemen, Styleguides oder Barrierefreiheitsstandards entsprechen [16].

Diese Automatisierung führt zu einem Paradigmenwechsel: Die Rolle des Designers verschiebt sich von der manuellen Erstellung hin zur Kuratierung und strategischen Steuerung der von der KI generierten Vorschläge [16].

5.3 Implementierung und Code-Generierung

Die offensichtlichste Auswirkung von LLMs liegt in der direkten Unterstützung bei der Programmierung. KI-Assistenten können Boilerplate-Code generieren, komplexe Algorithmen implementieren, Code übersetzen oder bestehenden Code refaktorisieren und dokumentieren. Dies führt zu einer signifikanten Steigerung der Entwicklerproduktivität.

Jedoch liegt die eigentliche Herausforderung nicht in der reinen Generierung, sondern in der Sicherstellung der Qualität und Sicherheit des erzeugten Codes, wie in Kapitel 4.2 diskutiert. Der "Human-in-the-Loop"-Ansatz ist hier unerlässlich: Der Entwickler muss die Vorschläge der KI kritisch prüfen, anpassen und die letztendliche Verantwortung für den Code übernehmen [10, 17].

5.4 Testen und Qualitätssicherung

Auch in der Phase des Testens bieten LLMs erhebliche Vorteile. Sie können auf Basis von Anforderungsdokumenten oder User Stories automatisch Testfälle generieren und so die Testabdeckung erhöhen [7]. Dies umfasst sowohl Unit-Tests zur Überprüfung einzelner Code-Komponenten als auch End-to-End-Tests, die komplette Nutzerflüsse simulieren.

Im Bereich des UX-Testings können KI-Systeme ebenfalls unterstützen, indem sie große Mengen an Nutzerfeedback aus Usability-Tests oder App-Store-Bewertungen analysieren und die häufigsten Usability-Probleme identifizieren. Prädiktive Modelle können sogar auf Basis eines UI-Designs vorhersagen, wo Nutzer potenzielle Schwierigkeiten haben könnten, und so bereits vor dem ersten Test wertvolle Hinweise zur Optimierung liefern [16].

6 Diskussion und Fazit

Die vorliegende Arbeit hat die vielfältigen Auswirkungen von "Large Language Models" (LLMs) auf den Softwareentwicklungsprozess beleuchtet und gezeigt, dass deren Einfluss weit über die reine Codegenerierung hinausgeht. Die Integration von KI-Technologien in den gesamten Lebenszyklus der Softwareentwicklung – von der Anforderungsanalyse bis zum Testen – markiert einen Paradigmenwechsel, der sowohl enorme Chancen als auch signifikante Risiken birgt.

6.1 Synthese der Erkenntnisse

Die Analyse hat ergeben, dass die größten Potenziale von LLMs in ihrer Fähigkeit liegen, als kollaborative Partner zu agieren. Sie beschleunigen den Design- und Forschungsprozess durch die Erstellung datengesteuerter Personas, die Automatisierung von Prototyping und die Analyse großer Datenmengen [16]. Im Idealfall ermöglichen sie eine "Mensch-Computer-Co-Kreativität", in der sich Entwickler auf übergeordnete, strategische Entscheidungen konzentrieren können, während die KI bei der Ausarbeitung unterstützt [9].

Diesen Chancen stehen jedoch gravierende Herausforderungen gegenüber. Das**"Black-Box"-Problem** untergräbt das Vertrauen und erschwert die Fehlersuche, während die unkritische Übernahme von KI-generiertem Code, wie die Studie von Perry et al. (2023) eindrücklich belegt, zu **signifikanten Sicherheitslücken** führen kann [10]. Die aktuellen Benchmarks, obwohl sie Fortschritte bei der Bewertung der Coding-Fähigkeiten machen, vernachlässigen qualitative Aspekte wie Sicherheit, Wartbarkeit und die Qualität der Mensch-KI-Interaktion [7, 9, 10].

6.2 Diskussion: Die neue Rolle des Softwareentwicklers

Die zentrale Frage, die sich aus diesen Erkenntnissen ergibt, lautet nicht, *ob* KI den Softwareentwickler ersetzen kann, sondern *wie* sie seine Rolle verändert. Die Antwort auf die Frage "Kann KI die Aufgaben eines Softwareentwicklers vollständig und fehlerfrei erledigen?" lautet nach aktuellem Stand eindeutig **Nein**. KI-Systeme können zwar Aufgaben effizienter und effektiver machen, doch die ultimative Verantwortung und Kontrolle muss beim Menschen bleiben [17].

Dies führt zu einer fundamentalen Verschiebung der erforderlichen Fähigkeiten:

- **Kritische Verifikation als oberstes Gebot:** Die wichtigste Fähigkeit ist nicht mehr nur das Schreiben von Code, sondern dessen kritische Überprüfung. Entwickler müssen in der Lage sein, die Vorschläge einer KI auf Korrektheit, Effizienz und vor allem Sicherheit zu validieren.
- **Meisterung des Prompt Engineering:** Die Qualität des KI-Outputs hängt entscheidend von der Qualität des Inputs ab. Die Fähigkeit, präzise und kontextbezogene Anweisungen (Prompts) zu formulieren, wird zu einer Kernkompetenz.
- **Fokus auf übergeordnete Fähigkeiten:** Anstatt sich in Implementierungsdetails zu verlieren, können und müssen sich Entwickler stärker auf die Systemarchitektur, das User Experience Design und die strategische Problemlösung konzentrieren.

Die Etablierung eines konsequenten**"Human-in-the-Loop"-Ansatzes** ist daher nicht nur eine Empfehlung, sondern eine Notwendigkeit, um die Potenziale der KI verantwortungsvoll zu nutzen und die Risiken zu minimieren [17].

6.3 Fazit

"Large Language Models" sind keine magischen Werkzeuge, die fehlerfreien und perfekten Code auf Knopfdruck liefern. Sie sind vielmehr extrem leistungsfähige Assistenzsysteme, deren effektiver Einsatz ein hohes Maß an Fachwissen, kritischem Denken und Verantwortungsbewusstsein erfordert.

Die Zukunft der Softwareentwicklung liegt nicht in der vollständigen Automatisierung durch KI, sondern in einer symbiotischen Zusammenarbeit zwischen Mensch und Maschine. Wenn Entwickler lernen, die Stärken von LLMs gezielt zu nutzen und gleichzeitig deren Schwächen durch menschliche Expertise und Kontrolle auszugleichen, kann dies zu einer erheblichen Steigerung von Effizienz, Effektivität und letztendlich der Qualität in der Entwicklung komplexer Softwaresysteme führen. Der richtige Einsatz

von LLMs wird somit zu einer Schlüsselkompetenz für die nächste Generation von Softwareentwicklern.

Acknowledgments

Mein Dank gilt Prof. Dr. Bastian Pfleging für die Betreuung dieser Arbeit.

References

[1] Reem Aleithan, Haoran Xue, Mohammad Mahdi Mohajer, Elijah Nnorom, Gias Uddin, and Song Wang. 2024. SWE-Bench+: Enhanced Coding Benchmark for LLMs. *arXiv preprint arXiv:2410.06992* (2024).

[2] Analytics Vidhya. 2025. Top LLM Benchmarks to Evaluate the Performance of Large Language Models. Abgerufen am 28. August 2025 von https://www.analyticsvidhya.com/blog/2025/03/llm-benchmarks/.

[3] Artificial Analysis. 2025. AI Model & API Provider Comparisons. Abgerufen am 25. August 2025 von https://artificialanalysis.ai/.

[4] Cloudflare. 2025. Human in the Loop · Cloudflare Agents docs. https://developers.cloudflare.com/agents/concepts/human-in-the-loop/

[5] Barnacle Goose. 2024. DeepSeek's new R1-0528 Performance Analysis and Benchmark Comparisons. Abgerufen am 28. August 2025 von https://medium.com/@leucopsis/deepseeks-new-r1-0528-performance-analysis-and-benchmark-comparisons-6440eac858d6.

[6] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations (ICLR)*.

[7] Naman Jain, King Han, Alex Gu, Fanjia Yan, Wen-Ding Li, Tianjun Zhang, Sida I. Wang, Koushik Sen, Ion Stoica, and Armando Solar-Lezama. 2024. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code. *arXiv preprint arXiv:2403.07974* (2024).

[8] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770* (2023).

[9] Michael Muller, Lydia B. Chilton, Anna Kantosalo, Mary Lou Maher, Charles Patrick Martin, and Greg Walsh. 2022. GenAICHI: Generative AI and HCI. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts (CHI '22 Extended Abstracts)*. ACM, New Orleans, LA, USA, 1–7. doi:10.1145/3491101.3503719

[10] Neil Perry, Megha Srivastava, Deepak Kumar, and Dan Boneh. 2023. Do Users Write More Insecure Code with AI Assistants?. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*. ACM, Copenhagen, Denmark, 2785–2799. doi:10.1145/3576915.3623157

[11] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julian Michael, Julien Dirani, and Samuel R. Bowman. 2023. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. *arXiv preprint arXiv:2311.12022* (2023).

[12] SWE-bench Team. 2025. SWE-bench Leaderboard. Abgerufen am 28. August 2025 von https://www.swebench.com/.

[13] Vals.ai. 2025. SWE-bench Benchmark. Abgerufen am 28. August 2025 von https://www.vals.ai/benchmarks/swebench-2025-08-27.

[14] Varad Vishwarupe, Shrey Maheshwari, Aseem Deshmukh, Shweta Mhaisalkar, Prachi M. Joshi, and Nicole Mathias. 2022. Bringing Humans at the Epicenter of Artificial Intelligence: A Confluence of AI, HCI and Human Centered Computing, In International Conference on Industry Sciences and Computer Science Innovation. *Procedia Computer Science* 204, 914–921. doi:10.1016/j.procs.2022.08.111

[15] Worldline. 2021. Ever heard of the AI black box problem? https://worldline.com/en/home/main-navigation/resources/blogs/2021/ever-heard-of-the-ai-black-box-problem.html

[16] Wei Xu. 2023. AI in HCI Design and User Experience. In *Human Computer Interaction: Interacting in Intelligent Environments*, Constantine Stephanidis and Gavriel Salvendy (Eds.). CRC Press, Boca Raton, FL, Chapter 5, 1–30. Preprint available as arXiv:2301.00987.

[17] Wei Xu, Marvin J. Dainoff, Liezhong Ge, and Zaifeng Gao. 2023. Transitioning to Human Interaction with AI Systems: New Challenges and Opportunities for HCI Professionals to Enable Human-Centered AI. *International Journal of Human-Computer Interaction* 39, 3 (2023), 494–518. doi:10.1080/10447318.2022.2041900

[18] Yueting Zhang, Arzoo Atiq, and Winn Chow. 2024. Exploring the Role of AI in UX Research: Challenges, Opportunities, and Educational Implications. In *Proceedings ASCILITE 2024*, T. Cochrane, V. Narayan, E. Bone, C. Deneen, M. Saligari, K. Tregloan, and R. Vanderburg (Eds.). ASCILITE, Melbourne, Australia, 556–560. doi:10.14742/apubs.2024.1341

[19] Zilliz. 2024. What is MMLU Benchmark? Abgerufen am 28. August 2025 von https://zilliz.com/glossary/mmlu-benchmark.