www.i…m.de

Ausgabe 1/2024
D € 12,90 · A € 13,90 · CHF 20,20

4 194156 112901 01

JavaSPEKTRUM
Magazin für professionelle Entwicklung und digitale Transformation

Entwickeln mit Generative AI und ML

GitHub Copilot – ein Bericht aus der Praxis

Was bedeutet Generative AI für die Java-Welt?

Was will ich werden?

Einstieg in die Softwareentwicklung heute noch sinnvoll?

Entwickeln mit Generative AI und ML

G30325

Was will ich werden?

Einstieg in die Softwareentwicklung heute noch sinnvoll?

Jan Baumann, Steve Haupt

Stellen Sie sich vor, es ist das Jahr 2030. Sie arbeiten in der Softwareentwicklung, setzen sich an Ihren Schreibtisch, umgeben von flimmerndem Licht und dem sanften Summen der Server. Stand heute würden Sie damit beginnen, eine Zeile Code nach der anderen zu tippen. 2030 ist das Geschichte. Statt zu tippen, führen Sie ein intelligentes Gespräch mit Ihrer KI-Assistenz, die – im Gegensatz zu uns Menschen – auf besondere Weise bilingual ist: Sie kann mit Ihnen reden und mit den Maschinen. Willkommen in der Zukunft der Softwareentwicklung! Einer Zukunft, die vielleicht näher ist, als wir denken.

DALL·E 3 mit dem Prompt: „Ein Software Entwickler programmiert. Er blickt dazu auf einen übergroßen Bildschirm. Dort ist links eine Art Smartwatch als Drahtgitter zu sehen mit der Unterschrift "Product". Rechts ist Quellcode zu sehen mit der Unterschrift "IDE". In der Mitte ein holographischer Roboterkopf als Gesprächspartner mit der Unterschrift "AI Assistant". Der Entwickler ist von der Seite zu sehen, der Blinkwinkel ist etwas schräg unten. Die Szenerie ist freundlich und optimistisch."

Dass in der IT-Branche nur der Wandel Bestand hat, ist hinlänglich bekannt. Innovationen und neue Technologien sind alltäglich, das macht aus jedem Hype zunächst einen unter vielen. Was zu der Frage führt, wie es sich mit der KI verhält – was ist Aufregung und was hat Substanz? Dazu möchten wir in diesem Artikel einen fundierten Einblick in die aktuelle Landschaft der Softwareentwicklung geben. Wir beginnen mit einem Überblick über die typischen Tätigkeiten in diesem Bereich und stellen Ihnen einige der KI-Tools vor, die bereits heute den Arbeitsalltag erleichtern. Und dann werfen wir einen differenzierten Blick in die Zukunft: einerseits mit den Entwicklungen, die wir erwarten, andererseits mit Spekulationen darüber, was sich auf dem Boden der Kristallkugel noch so befinden könnte ...

Was Softwareentwickler so machen

Wir gehen in diesem Artikel von zwei Prämissen aus. Unsere erste Annahme lautet, dass Agile Software-Engineering (ASE) in den meisten Produkten und Projekten praktiziert wird. Gemäß unserer zweiten Prämisse profitiert ASE davon, die Entwickler so früh wie möglich in die Definition der Anforderungen einzubinden.

Softwareentwicklung beginnt mit einem fachlichen Problem, sei es real oder vermutet. Einer der ersten Schritte besteht folglich darin, für ein fachliches Problem eine fachliche Lösung zu finden. Das ist naturgemäß die Aufgabe der fachlichen Experten, die Entwickler sind daran, je nach Unternehmen, mal mehr, mal weniger beteiligt. Im Idealfall ist diese Beteiligung intensiv, denn besagte fachliche Lösung soll mit der Entwicklung technisch umgesetzt werden. Meist gehört dazu, die Lösung in ein bestehendes System zu integrieren.

Damit besteht die eigentliche Aufgabe nicht primär darin, neuen Code zu schreiben. Sie umfasst ebenso, bestehenden Code zu lesen und zu verstehen, was oft mehr Zeit in Anspruch nimmt als gedacht [Mar09]. Um für die geforderte Wartbarkeit zu sorgen, sind dabei auch geeignete Maßnahmen wie Dokumentation und Code Reviews notwendig. Natürlich muss auch die Funktionalität sichergestellt werden, damit kommen nicht-funktionale Anforderungen wie Performance, Ausfallverhalten und Benutzbarkeit zum Aufgabenkatalog hinzu. Dafür sind geeignete Tests und, basierend auf den Ergebnissen, Korrekturen erforderlich.

Typischerweise kümmern sich Entwickler auch um ihren Entwicklungsprozess und dessen Verbesserung und um die Kommunikation über die Teamgrenzen hinweg. Das betrifft zum Beispiel die Betreuung der Kunden und Anwender oder die Abstimmung mit anderen Entwicklungsteams. Damit addieren sich die Tätigkeiten zu dieser Liste:

- Anforderungserhebung und -management,
- Lesen und Verstehen von Code,
- Schreiben von Code,
- Sicherstellen der Wartbarkeit,
- Sicherstellen der Funktionalität,
- Team-externe Kommunikation.

Die Liste deckt, unserer Erfahrung nach, die typischen Tätigkeiten in der Softwareentwicklung auf einem hohen Abstraktionsniveau ab. Für die folgende Betrachtung muss sie weder vollständig detailliert noch überschneidungsfrei unterteilt sein. Die Liste reicht bereits aus für eine Diskussion darüber, wo die KI-Systeme aktuell stehen und wo ihre Grenzen liegen.

Jan Baumann arbeitet seit 2008 in der professionellen Softwareentwicklung bei der andrena objects ag. Er liebt es, Produkte zu entwickeln und durfte das schon in den verschiedensten Rollen ausgiebig tun – angefangen als Entwickler und Trainer, dann mit Abstechern ins Product Ownership und der Architektur bis hin zu seiner aktuellen Tätigkeit als Scrum Master. Er lernt trotzdem fleißig weiter neue Rollen, Fachlichkeiten und Technologien kennen – einer faszinierenden Entwicklung wie dem Aufstieg der generativen KI konnte er gar nicht widerstehen.
E-Mail: jan.baumann@andrena.de

Steve Haupt arbeitet als agiler Softwareentwickler bei andrena. Er betrachtet Softwareentwicklung als Handwerk, das über einen längeren Zeitraum erlernt und geübt werden muss. Besonders fasziniert ist Steve von den aktuellen Entwicklungen im Bereich der KI und reflektiert intensiv über die Implikationen dieser Technologien für das Handwerk der Softwareentwicklung.
E-Mail: steve.haupt@andrena.de

State of the Art 2024

Seit der Veröffentlichung von ChatGPT hat die Integration von KI-Systemen in den Arbeitsalltag rasant zugenommen. Die ersten KI-gestützten Werkzeuge, die aus den Forschungslaboren in die alltägliche Praxis der Softwareentwicklung übergegangen sind, sind sogenannte Coding-Assistenten. Sie sind in der Entwicklungsumgebung (IDE) direkt eingebunden und unterstützen uns beim Schreiben von Code, indem sie versuchen, den nächsten Code-Schnipsel vorherzusagen.

Das ist nichts bahnbrechend Neues, so können professionelle IDEs schon lange intelligente Vorschläge machen, ganz ohne KI. Zum Beispiel, wenn man in einer Java-Umgebung

```
ArrayList<String> myList = new A
```

tippt, erkennt die IDE automatisch, was man vorhat, und vervollständigt den Code zu

```
ArrayList<String> myList = new ArrayList<>();}
```

Ein einfacher Tastendruck, und der vorgeschlagene Code wird akzeptiert. Doch KI-basierte Coding-Assistenten wie GitHub Copilot gehen weit darüber hinaus. Sie nutzen Large Language Models (LLMs) zur Generierung umfangreicherer Codefragmente. Zum Beispiel könnten sie in einem Spring Boot-Projekt eine vollständige Implementierung einer REST-API-Methode vorschlagen, sobald Sie die Anfangszeilen für eine Controller-Klasse getippt haben. Oder einen Unittest vervollständigen, während man noch die Methodensignatur tippt.

Laut einer Stack Overflow-Umfrage wollen 75 Prozent der befragten Entwickler GitHub Copilot einsetzen. Von denjenigen, die Copilot bereits verwenden, wollen 75 Prozent damit weitermachen [Stack0]. Andere, nicht so bekannte Coding-Assistenten sind AWS CodeWhisperer, Tabnine Whipser AI, Replit Ghostwriter, Rubber Duck.AI, Synk Code, Codeium, Adrenaline und Mintlify. Was die Qualität der Vorschläge angeht, hängen sie GitHub Copilot hinterher, dafür können viele davon ohne Cloud-Anbindung laufen, was in Softwareprojekten häufig verlangt wird.

Zusätzlich zu Coding-Assistenten haben sich LLMs auch als hervorragendes Werkzeug etabliert, um Beiträge zu fachlichen Themen schnell aufzufinden. War es früher oft notwendig, zahlreiche Foren und Blogs zu durchforsten, liefern KI-Bots heute präzise Antworten und stehen für Rückfragen zur Verfügung.

Dieser technologische Fortschritt macht den Beruf der Softwareentwicklung besonders attraktiv für Quereinsteiger. Denn grundsätzlich müssen Entwickler über zwei Dinge verfügen: über ein ausgeprägtes abstraktes Denkvermögen und Fachwissen. Das abstrakte Denkvermögen ist eine grundlegende Fähigkeit, welche in vielen Fachrichtungen trainiert wird. Das Fachwissen (Syntax, Technologien, Patterns, Best Practices ...) kann – oder besser gesagt muss – mühsam über einen langen Zeitraum hinweg erworben werden. Dank KI-Systemen wie ChatGPT ist dieses Fachwissen nun leichter denn je über einen natürlichen Dialog zugänglich, was einen wesentlich schnelleren Zuwachs der Kompetenz ermöglicht.

Reine Chat-Anwendungen wie ChatGPT, Google Bard und Claude sind ein guter Startpunkt. Dennoch halluzinieren sie noch häufig, können nur eingeschränkt auf aktuelle Informationen zugreifen und in der Regel Quellen nicht sauber zitieren. Die KI-Suchmaschine Phind geht einen Schritt weiter. Sie ist für die Softwareentwicklung optimiert, bietet mehrere konfigurierbare Suchmodi, liefert Quellen und kann selbstverständlich auch in den Chat-Modus wechseln.

Ein weiterer Trend, der heute gut zu beobachten ist, besteht in der Automatisierung kleiner, unbeliebter Tasks, die im Entwickleralltag häufig anfallen. So hat der IDE-Hersteller JetBrains zum Beispiel mit der Version 2023.2 den „AI Assistant“ vorgestellt, der initial nur über eine Warteliste aktivierbar ist, dann aber zusätzlich zu einem Chat auch Dokumentation für Code, Namensvorschläge und Commit-Nachrichten generieren kann. Andere Anwendungen wie CodeRabbit oder Codium spezialisieren sich auf die automatisierte Analyse von Pull Requests oder die Testgenerierung.

Der Motor, der alle bisher beschriebenen KI-Werkzeuge antreibt, sind Sprachmodelle wie GPT-4. Sie werden mit jeder neuen Iteration fähiger und werden mehr und mehr den Kontext verstehen und immer hochwertigere Vorschläge generieren.

Der Blick in die Kristallkugel

Hier beginnen wir nun, zu spekulieren – oder, um genauer zu sein, uns in die Spekulationen einzuklinken, die schon seit einiger Zeit im Internet zu lesen sind. Als Ausgangspunkt nehmen wir dabei einen sehr lesenswerten Beitrag: den Artikel „Are developers needed in the age of AI“ des Agile Coach Henrik Kniberg [Kni23]. In dieser Veröffentlichung zeichnet er die folgenden fünf Phasen:

- KI-Assistenten unterstützen das Entwicklungsteam,
- KI ist ein Mitglied des Entwicklungsteams,
- KI/Entwickler-Tandems sind das ganze Entwicklungsteam,
- KI ersetzt das Entwicklungsteam vollständig,
- KI generiert direkt Maschinensprache.

Phase #1 haben wir bereits in dem Blick auf den heutigen Arbeitsalltag mit KI-Tools beleuchtet – hier geht es vor allem um intelligente Assistenten für die Entwickler, die dafür sorgen, dass

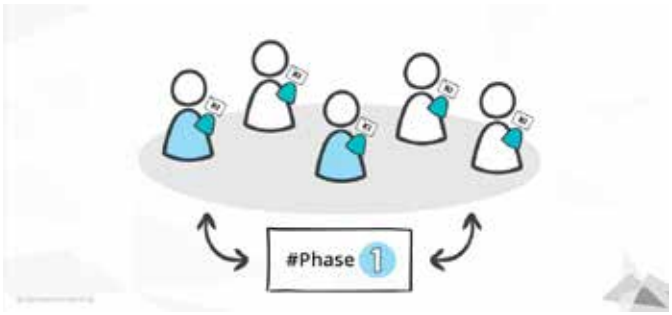

Abb. 1: KI-Assistenten unterstützen das Entwicklungsteam

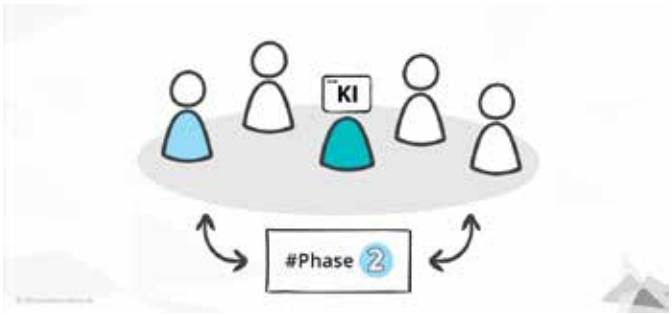

Abb. 2: KI ist ein Mitglied des Entwicklungsteams

die Effizienz und der Erfahrungsgrad steigen. Wenden wir uns nun also den weiteren Szenarien zu.

In Phase #2 ist die KI ein vollwertiges Teammitglied. Sie ist Generalist mit gewissen Grenzen, wie das für viele Teams in der Softwareentwicklung gilt. Da insbesondere das Thema Anforderungserhebung eine große Breite haben kann und beim Thema Prozess- und Produktsteuerung meist eine besonders gute Kommunikation notwendig ist, könnte ein beispielhaftes Szenario sein, dass sich eine solche KI speziell auf die Themen Clean Code und Team-Coaching konzentriert. In so einem Szenario übernehmen einstweilen die menschlichen Kollegen noch Dinge wie die Identifikation der Anforderungen sowie die Prozess- und Produktsteuerung.

Nötig wäre in diesem Szenario eine Verbindung verschiedener bestehender Expertensysteme, angefangen mit Speech-to-Text und Text-to-Speech, damit die KI die Möglichkeit hätte, sich virtuell an Meetings zu beteiligen und kurzfristig Rückfragen zu stellen. Durch einen Zugriff auf alle nötigen Entwicklungssysteme könnte die KI gezielte Vorschläge zu möglichen Problemen machen, um so ihrer Rolle im Team gerecht zu werden. Spätestens jetzt werden viele Menschen die KI als direkte Konkurrenz empfinden, was sie de facto auch ist, und die Frage aufwirft, inwieweit Menschen bereit sind, diese Konkurrenz zu akzeptieren. Zumal man einer KI nicht so leicht vergibt wie einem Menschen (vgl. [Hid21]). Denkbar wäre, dass die KI ab hier jedoch Schritt für Schritt Teammitglieder ersetzt, wodurch man schließlich in Phase #3 ankommt.

In Phase #3 übernimmt die KI einen Großteil der Arbeiten des früheren Entwicklungsteams, wie Kniberg sagt, hat sie ja praktisch „unendliche“ Kapazität. Dafür sind im Vergleich zu Phase #2 in unseren Augen hauptsächlich inkrementelle Verbesserungen nötig, da es ja immer noch einen menschlichen Partner gibt, der bei schwierigen Themen wie Anforderungsanalyse, Teamkoordination und Prozesssteuerung unterstützen kann. Gleichzeitig wird dieser Mensch dadurch schnell zu einem Engpass, welcher die Leistungsfähigkeit der KI beschränkt. An dieser Stelle ist das Akzeptanzproblem geringer, denn aus Sicht der Kritiker ist das Kind ja bereits in den Brunnen gefallen und die Kritiker selbst sind (wahrscheinlich) nicht mehr da.

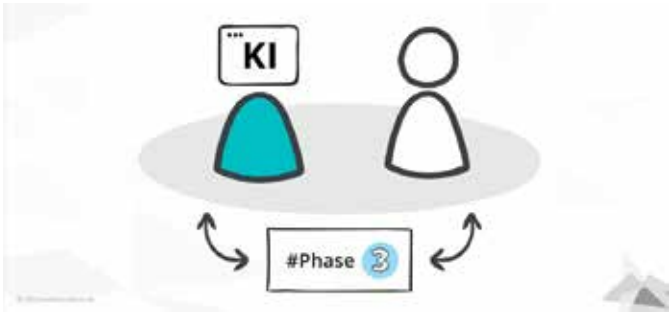

Abb. 3: KI/Entwickler-Tandems sind das ganze Entwicklungsteam

Was stattdessen zunehmend an Bedeutung gewinnt, wird das Vertrauen in die KI-Ergebnisse sein, denn der menschliche Partner wird längst nicht mehr alles überprüfen können, was die KI macht. Ebenso wird aus Sicht des Risikomanagements wichtiger, sich zu überlegen, wie man das eine KI-System, von dem nun der gesamte Betrieb abhängt, geeignet absichert, um nicht plötzlich vor dem Nichts zu stehen. Hat man diese Hürden jedoch gemeistert, sind wir erfolgreich in der Vision, die wir zum Artikelanfang skizziert haben. Aber dort hört Kniberg noch nicht auf – weiter zu Phase #4.

Hier, in Phase #4, interagiert der fachliche Anwender direkt mit der KI. Der größte Unterschied ist nun, dass der fachliche Anforderer gar nicht mehr mit einem Menschen spricht. Denn bisher gab es noch mindestens eine Person, die bei beiden Schritten dabei war: bei der Übersetzung des fachlichen Problems in eine fachliche Lösung und, gemeinsam mit der KI, beim Übertragen dieser fachlichen Lösung in eine technische Funktionalität. Jetzt wendet sich der fachliche Anwender mit seinem fachlichen Problem direkt an die KI, die ihm dann eine technische Lösung präsentiert.

Für uns stellt das noch mal einen riesigen Sprung dar. Denn um eine fachliche Lösung zu finden, braucht es eine Vielzahl von Ideen, unterschiedliche Hypothesen müssen getestet und verschiedene Dinge ausprobiert werden. Von der rein technischen Umsetzung ist da noch gar nicht die Rede. Zusätzlich zu den rein technischen Aspekten der vorigen Phasen müsste sich die KI darüber hinaus also auch deutlich tiefer in der fachlichen Welt, der Politik der Organisation, der organisatorischen Prozesse und den vielen, typisch menschlichen Problemen in großen sozialen Systemen auskennen. Vermutlich werden spätestens dann KI-Systeme den sagenumwobenen Status AGI (Artificial General Intelligence) erhalten haben. Für den fachlichen Anforderer, der Stand heute die technischen Aspekte der entwickelten Software nicht versteht, verschiebt sich dann lediglich das Vertrauen vom ehemaligen Entwicklungsteam hin zur KI.

Damit kommen wir zu einem neuen, interessanten Problem, das dagegensprechen könnte, dass es je so weit kommt: Was ist, wenn die KI dann doch irgendwann an ihre Grenzen stößt? Es gibt nun überhaupt keinen Menschen mehr, der das System auch nur annähernd verstehen würde. Das betriebliche Risiko ist hier noch einmal höher als in Phase #3.

Was ändert sich dann noch in Phase #5? Da ohnehin kein Mensch mehr den Quellcode kennt, und es utopisch wäre, Systeme mit Millionen Zeilen von Code schnell genug verstehen zu wollen, um bei Problemen sinnvoll zu helfen, könnte der Quellcode einfach direkt wegfallen. Die KI kann direkt die Maschine instruieren. Der Unterschied dürfte sich, was praktische Belange angeht, in Grenzen halten. Wir denken, falls wir je zu Phase #4 kommen, wird Phase #5 nicht mehr lange auf sich warten lassen.

Allen Phasen gemein ist, dass man stets Kosten und Nutzen betrachten muss. Schon in Phase 2 muss die KI ständig „weiterlernen“, diese Anforderung steigt kontinuierlich mit jeder dazukommenden Fähigkeit. Dieses „Weiterlernen“ kann gelöst werden durch das Nachtrainieren der KI oder durch die Anreicherung des Kontexts, den die KI bekommt. Das klassische Nachtrainieren (updaten der Parameter) ergibt in den allermeisten Fällen keinen Sinn, da heutige KI-Systeme sehr ineffizient lernen, das heißt, sehr viele Trainingsdaten brauchen, die in den meisten Fällen nicht vorliegen. Die Anreicherung des Kontexts ist eine elegante Methode, um dieses Problem zu umgehen, sie ist allerdings technisch begrenzt auf eine maximale Kontextgröße. Doch selbst wenn diese Grenze immer weiter verschoben wird, bedeutet dies, dass jede Anfrage, die an die KI geht, größer und größer wird, das heißt, mehr Ressourcen benötigt.

Einen entscheidenden Einfluss hat auch die Fehlerrate der KI-Systeme. Ähnlich wie beim autonomen Fahren muss die Frage beantwortet werden, wann die Fehlerraten klein genug sind, oder anders ausgedrückt, wann die Systeme für den jeweiligen Use Case gut genug sind. Es wird spannend sein, wie die geringer werdende Fehlerrate im Vergleich zur Effizienzsteigerung ausfällt. Dazu kommen ökologische Betrachtungen durch den Betrieb der Hardware und der KI sowie rechtliche Fragen rund um das Thema Urheberrecht (aktuell unterliegen KI erzeugte Texte nicht dem Urheberrecht [UHR]) und Datenschutz.

Lohnen könnte sich der Einsatz überall dort, wo Fehler nicht unternehmenskritisch sind – etwa in vielen Apps, die primär der Freizeit dienen. Auch in einem Umfeld, das von einer experimentierfreudigen und innovationsoffenen Haltung geprägt ist, etwa in Start-ups, könnte er sich lohnen. Zudem wäre der Einsatz in Situationen denkbar, in denen rechtliche Bedenken nur eine untergeordnete Rolle spielen.

Die Quintessenz

Zusammenfassend lässt sich sagen: Die ASE befindet sich auf dem Pfad der Automatisierung. Aktuell stehen wir am Anfang und die zwei unbekannten Parameter Geschwindigkeit (langsam/schnell) und Grad der Automatisierung (teilweise/vollständig) lassen sich Stand heute noch nicht abschätzen. Deswegen finden wir es voreilig, manche Kombinationen (wie schnelle und vollständige Automatisierung) auszuschließen. Zumindest in Gedanken möchten wir dazu ermutigen, solche Szenarien zuzulassen und durchzuspielen. Denn wie immer in der ASE gilt: *Plans are worthless but planning is essential*.

Für Berufsanfänger und -einsteiger drängt sich damit eine Frage in den Mittelpunkt: Ist der Einstieg in die Softwareentwicklung heute noch sinnvoll? Der Blick auf die unmittelbare Zukunft zeigt – ja sicher, es gab noch nie eine bessere Zeit, um in die ASE einzusteigen! KI-Tools erleichtern den Einstieg wie noch nie zuvor. Sie greifen unter die Arme und helfen bei Fragen zur Syntax oder Fehlermeldungen. Außerdem sorgen sie dafür, dass wir uns mehr und mehr auf komplexere und intellektuell ansprechende Probleme konzentrieren können. Träge, langweilige, stupide Aufgaben (z. B. Boilerplate-Code) halten sie uns vom Hals.

Doch was, wenn die Automatisierung schneller und radikaler geschieht, als es sich die meisten von uns vorstellen können? Selbst dann gibt es hervorragende Gründe, in die ASE einzusteigen. Zum einen ist da die prinzipielle Beobachtung, dass uns die Arbeit trotz mehrfacher Ankündigung und mehreren industriellen Revolutionen noch nicht ausgegangen ist. Sie hat sich jeweils stark verändert, doch die Agile Softwareentwicklung betont deswegen die Agilität, weil es in ihr nur eine Konstante gibt – und das ist die Veränderung. Veränderung an der Software, an den Anforderungen, im Team, an der Technologie, in den Best Practices und an unserer Arbeitsweise. Veränderung und die Anpassung daran ist selbstverständlicher Bestandteil unseres Arbeitsalltags. Damit haben wir vermutlich auch langfristig sehr gute Voraussetzungen, um mit der Veränderung durch KI umgehen zu können.

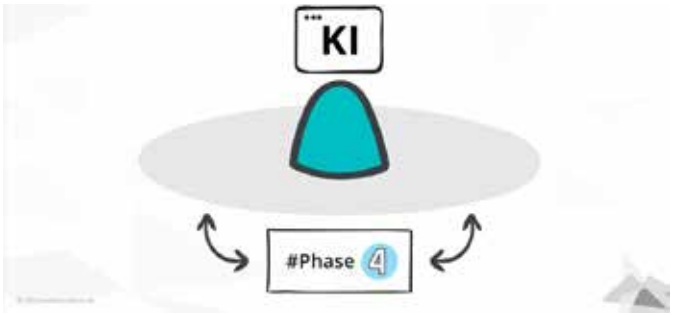

Abb. 4: KI ersetzt das Entwicklungsteam vollständig

Mit einer Ausbildung als ASE haben wir uns zudem darin geschult, kreative Lösungen zu finden und auf einem hohen Abstraktionsniveau zu denken. Außerdem haben wir wichtige Fähigkeiten über Computersysteme, Vernetzung und technische wie fachliche Komplexität gesammelt. Viele Entwickler sind in ihren fachlichen Kenntnissen fast auf dem Niveau ihrer Anforderer. All das sind Fähigkeiten, die uns in Zusammenarbeit mit der KI weiterhelfen können und uns so zu wertvollen KI-Mitarbeitern macht [McN23].

Zum anderen ist zu erwarten, dass es immer Bereiche geben wird, in denen der Einsatz von KI nicht sinnvoll oder möglich ist – so wie es heute ja trotz allem Nutzen auch Bereiche gibt, wo noch ohne Computer gearbeitet wird, oder so wie es lange Zeit immer noch Positionen für Fortran- oder Cobol-Entwickler gab. Für diese Einsatzgebiete wird es weiterhin ASE brauchen.

Was könnt Ihr also jetzt tun? Probiert Dinge aus, tauscht Euch mit Kollegen aus. Bildet Arbeitsgruppen. Nutzt Research-Tage für KI-Hackathons. Jetzt ist die Zeit, sich mit der Technologie zu beschäftigen, welche unsere Zukunft maßgeblich beeinflussen wird! Das Gute dabei ist: Mit LLMs zu spielen, das macht unglaublich Spaß!

Literatur und Links

[Hid21] C. A. Hidalgo, Why we forgive humans more readily than machines, Sep. 2021, www.scientificamerican.com/article/why-we-forgive-humans-more-readily-than-machines/

[Kni23] H. Kniberg, Are developers needed in the age of AI?, Hups AB, 25.4.2023, hups.com/blog/are-developers-needed-in-the-age-of-ai

[Mar09] R. C. Martin, Clean Code: Refactoring, Patterns, Testen und Techniken für sauberen Code, mitp-Verlag, 2009, S. 41 f.

[McN23] M. McNeilly, 20.2.2023, kenaninstitute.unc.edu/commentary/the-must-have-skills-in-the-era-of-artificial-intelligence-how-ais-democratization-will-impact-workers/

[Stack0] Stack Overflow. Admired and desired - ai developers tool, 2023, survey.stackoverflow.co/2023/#section-admired-and-desired-ai-developer-tools

[UHR] Künstliche Intelligenz (KI) im Urheberrecht: Welche Rechte bestehen?, 21.8.2023, www.urheberrecht.de/kuenstliche-intelligenz/