KI in der Softwareentwicklung

Besser, schneller, weiter

Über mich

Andriy Golovko
Dipl.-Inf. Univ. (TU München)
https://aagolovko.github.io/
golovko@mytum.de
+49 179 219 00 94

Inhalt

Motivation

Wie kann man KI-Unterstützung für Softwareentwicklung holistisch und souverän umsetzen? Hier will ich dieses Thema angehen und vorstellen, wie ich das umsetzen würde.

Mehrwert für Unternehmen

Warum soll es für ein Unternehmen (oder auch eine einzelne Person bzw. ein kleines Team) interessant sein, KI überhaupt für Softwareentwicklung zu verwenden? Die Antwort mag trivial sein: besser, schneller, weiter … Aber ich rekapituliere diese Punkte hier noch einmal, weil sie die Grundlage für messbaren Business-Impact bieten. Der Ansatz mit KI ist wichtig für:

- Weniger Rework, mehr Qualität
- Geringere technische Schulden
- Höhere Lieferfähigkeit und Planbarkeit

Leitfaden

Heutzutage gibt es ganz unterschiedliche Ansätze, wie man Softwareentwicklung mit KI unterstützen kann. Wie kann man aber diese Ansätze unter einem Dach zusammenbringen?

Der Begriff **Software Development Life Cycle** entstand im Wesentlichen in den 1950er und 1960er Jahren. Der SDLC wurde eingeführt, um die zunehmende Komplexität moderner Softwareprojekte durch methodische Strukturierung beherrschbar zu machen und so Qualität, Zeitplan und Budget abzusichern. Dieser Begriff ist fundamental für die Softwareentwicklung. Durch den Einsatz von KI lässt sich die Effizienz und Qualität im SDLC möglicherweise **auf ein neues Niveau** heben.

Der SDLC hat folgende Phasen: **Planung**, **Analyse**, **Design**, **Implementierung**, **Test**, **Deployment**, **Betrieb**. Alles, was mit Softwareentwicklung zu tun hat, dreht sich um den SDLC. Wir wollen alle diese Phasen so weit sinnvoll wie möglich **mit KI unterstützen**.

Herausforderungen und Risiken

Selbst wenn heutzutage viel über KI gesprochen und experimentiert wird, finde ich das Wort **„möglicherweise"** wichtig – vor allem in einem konkreten Unternehmen. Nach meinem Eindruck befinden wir uns derzeit in einer Pionierphase für KI-Ansätze. So wie es auch für Automobile einmal war, zwischen den 1880er und etwa 1910er Jahren: Nicht alle Ansätze werden überleben. Der Ansatz mit KI bietet natürlich Vorteile, ist aber auch mit gewissen Herausforderungen verbunden. Ich liste hier die auf, die meiner Meinung nach am wichtigsten sind.

Das erste Risiko liegt in meinen Augen im **Produktivitätsparadoxon**. Wer einen schlechten Entwicklungsprozess mit KI beschleunigt, beschleunigt seine technischen Schulden. Studien zeigen, dass der wahre Wert erst entsteht, wenn die Arbeitsweise selbst infrage gestellt wird. Business Impact entsteht nicht durch die Installation von KI, sondern durch die Transformation der Arbeit durch KI. Erläutert auf einem Beispiel: Mit KI kann man schneller Bugberichte dokumentieren. Aber mit KI-gestützter proaktiver Qualitätssicherung (entsteht durch Transformation der Arbeitsprozesse) entstehen die Bugs erst gar nicht.

Die nächste Herausforderung liegt darin, die KI möglichst holistisch einzusetzen – genauer formuliert: **"Holistische Integration statt isolierter Tools"**. Ich lese, dass viele über Erfahrungen mit Claude Code für lokale Softwareentwicklung berichten. Ich plädiere aber dafür, KI in allen SDLC-Phasen zu verwenden. Das soll helfen den Kontextverlust zwischen Phasen zu vermeiden oder zumindest minimieren. Es sei angemerkt, dass Kontext- bzw. Übergabeverluste zwischen Phasen entstehen, auch im SDLC ohne KI-Ansatz, wenn die Dokumente aus der Design-Phase zur Coding-Phase über den Zaun geworfen werden. Das führt dazu, dass das Gericht am Ende nicht so schmeckt wie geplant, obwohl alle Zutaten drin sind. Mit holistischer KI beim SDLC wollen wir diesen Kontextverlust vermeiden: mit Orchestrierungstools wie n8n, Workflow-Agenten und Automatisierung.

Wenn wir vor allem für fachliche Themen ein LLM nutzen, fehlen dort logischerweise die Kenntnisse, die ein Unternehmen über Jahre gesammelt hat und die es von anderen unterscheiden. Über RAG kann man interne Quellen als Wissenstreibstoff für die KI nutzen: transkribierte Team-Meetings, interne Confluence-Dokumentation, interne Laufwerke etc.

Damit sind offensichtlich Risiken verbunden. Und nach dem EU AI Act ist das keine theoretische Frage mehr, sondern eine rechtliche Pflicht. Eine **eine souveräne KI-Infrastruktur** aufzubauen ist deshalb nicht nur technische Herausforderung, wo ein Unternehmen die Kompromisse finden muss

- verarbeiten wir die Daten in einer (laut Studien teurer) On-Premise-KI, oder
- dürfen die Daten zu externen, günstigen KI-Anbietern fließen?

Nun haben wir einige Risiken und Herausforderungen identifiziert, die Vorteile sind aber so verlockend, dass wir KI einsetzen wollen: holistisch, mit Workflow-Agenten und souveräner RAG. Wie gehen wir vor?

Manifest

Dafür nutzen wir ebenfalls den SDLC – warum sollen wir das Rad neu erfinden? Ich würde aber vorab einige Leitlinien formulieren, die zumindest ich in meiner Arbeit verfolgen würde:

- **„Think big, do small"**: Zielbild (Zielarchitektur) im Blick behalten, in kleinen Schritten vorgehen. Experimentieren statt langer Entscheidungsphasen. Bauchgefühl ist willkommen.
- **Kennzahlbasierte Entscheidungen** darüber, ob dieser oder jener Ansatz gut oder schlecht ist. Bauchgefühl ist verboten.
- **Human-in-the-loop („Menschen first, KI second").** Der Mensch bleibt an den entscheidenden Stellen im Prozess eingebunden – KI schlägt vor, der Mensch entscheidet. Das ist das Gegenprinzip zum vollautomatischen „KI entscheidet alles". Damit verbunden ist Verantwortung: Unsere KI-Implementierungen müssen ethisch vertretbar sein, die Privatsphäre der Nutzer respektieren und in ihren Auswirkungen verstanden werden. Wer KI einsetzt, muss auch steuern können, was sie tut.
- **Iterativ eine Roadmap/einen Blueprint für den KI-Ansatz entwickeln.** Sie ist eine Grundlage für die unternehmensweite Wissensverbreitung. Und auch eine Grundlage für wiederverwendbare Lösungen.
- **KI ist keine Wunderwaffe**. Wir sollen die KI nicht überschätzen. Wir brauchen ein Schutz gegen Halluzinationen. Wir müssen sehen könne, wenn KI an die Grenzen stößt. In etwa mit RAG, durch vergleich mit Realität (KI Prognosen für Aufwand/Story Points sind oft stark abweichend von der Realität etc.), oder durch Feedback aus CI/CD (die Tests schlagen für KI Code oft fehl).
- **Intent Engineering.** KI auch nicht gegen unpräzise Anforderungen helfen, schlechte Requirements sind sofort sichtbar. Nicht wer schneller promptet, sondern wer präziser formuliert, gewinnt.
- **Wir wollen uns unseren Weg nicht verbauen.** Der Markt ändert sich schnell, und deshalb ist es wichtig, die Möglichkeit zu haben, z. B. von n8n auf etwas anderes umzusteigen. Technisch gesehen ist das möglich, wenn beispielsweise JSON und kein proprietäres Format verwendet wird. Mit A2A, ACP wird diese Anforderung adressiert.
- **Phasengedächtnis**. Isolierte KI-Tools beginnen an jeder Phasengrenze bei null. Was in der Anforderungsphase erarbeitet wurde – Constraints, Domänensprache, priorisierte Use Cases – steht dem nächsten Agenten nicht zur Verfügung. Ein persistenter Kontext über alle SDLC-Phasen ist deshalb kein Nice-to-have, sondern die Grundvoraussetzung für holistisches Arbeiten.
- **Agentic Engineering.** Wer KI ohne Struktur einsetzt, betreibt Vibe Coding – und beschleunigt damit seinen technischen Schuldenberg. Agentic Engineering ist ein Gegenbegriff zu Vibe Coding. Wir setzen auf strukturiertes, intentionales Arbeiten mit KI-Agenten.
- **Mit KI darf man sich mehr gönnen.** KI ist wie ein Exoskelett für die Softwareentwicklung – es verstärkt, was man bereits kann. Ein Entwickler gewinnt dadurch Kapazität, um über die Strategie des Produkts nachzudenken, statt sich im Code zu verlieren. Man darf von sich selbst mehr erwarten.

Zielbild: gönne sich mehr

Mit KI darf man sich mehr gönnen – das ist der Leitgedanke, mit dem ich an das Thema herangehe. Gleichzeitig gilt: „Think big, do small". Deshalb strukturiere ich den KI-Einsatz entlang von Reifestufen. In diesem Artikel setze ich den Fokus auf Stufe 2: **Orchestrierung im SDLC**. Der Grund ist einfach: Die meisten Unternehmen haben diese Stufe noch nicht erreicht (meine Intuition). Sie ist aber machbar genug, um heute damit anzufangen, und anspruchsvoll genug, um den vollen Mehrwert von KI zu realisieren. Die Machbarkeit belege ich mit konkreten technischen Konzepten.

Vier Reifestufen der KI-Integration:

(1) Lokale Entwicklung – KI als persönliches Werkzeug. Claude Code, Cursor, Copilot. Der Einstieg, den viele bereits kennen.

(2) Orchestrierung im SDLC – KI-Agenten übernehmen Aufgaben über alle Entwicklungsphasen. Ein oder mehrere Frameworks arbeiten zusammen. Das ist der Kern dieses Vortrags.

(3) Unternehmensweite KI-Workflows – KI verlässt die Softwareentwicklung und unterstützt andere Geschäftsprozesse: HR, Finance, Operations. Der SDLC war der Proof of Concept.

(4) Agenten-Ökosystem – Unsere KI-Agenten kommunizieren mit denen anderer Unternehmen. Die technologische Basis existiert bereits: MCP, A2A. Die konkreten Szenarien sind noch offen – aber vermutlich näher als gedacht. Ich kann die Machbarkeit von Stufe 4 noch nicht vollständig beurteilen. Meine Verständnisse aber: Die Protokolle existieren. Was fehlt, ist die Erprobung.

Ab Stufe 2 wird **großer Wert auf KI-Agenten** gelegt. Warum eigentlich? Warum nicht alles in einem einzigen Agenten erledigen? Ich beantworte das mit drei Gedanken:

(a) Teile und herrsche als Fundament. Ein einzelner Prompt hat Kontextgrenzen – je länger und komplexer die Aufgabe, desto mehr leidet die Qualität. Agenten teilen die Komplexität in beherrschbare Einheiten. Jeder Agent bekommt genau den Kontext, den er braucht – nicht mehr, nicht weniger. Ein Agent, der nur Code-Reviews macht, ist mit einem spezialisierten Prompt besser kalibriert als ein Generalist-Prompt, der alles auf einmal soll. Wir wollen Kontextverlust vermeiden – aber wir wollen auch, dass jeder Agent nur den relevanten Ausschnitt bekommt, den er für seine Aufgabe braucht. Der Orchestrator entscheidet, welcher Agent welchen Kontext erhält. So vermeiden wir das „Lost in the Middle"-Problem: Informationen in der Mitte eines langen Kontexts werden schlechter verarbeitet als am Anfang oder Ende. Analogie: Ein Chirurg braucht die Patientenakte – aber nicht die Buchhaltung des Krankenhauses. Beides existiert, aber nur das Relevante landet auf dem Tisch.

(b) Unterschiedliche Blickwinkel als Qualitätsmechanismus. Gleicher Kontext, aber aus verschiedenen Perspektiven betrachtet. Als Analogie dient die „Six Thinking Hats"-Methode: Dieselbe Situation wird bewusst aus unterschiedlichen Blickwinkeln analysiert – um blinde Flecken zu vermeiden und zu besseren Entscheidungen zu kommen.

(c) Parallelität als Effizienzgewinn. Mehrere Agenten arbeiten gleichzeitig – was die Durchlaufzeit reduziert und den Gesamtprozess beschleunigt. Das ist aber eher ein technischer Vorteil als ein konzeptioneller.

Zielbild: (2) Orchestrierung im SDLC

Ich würde die Zielarchitektur basierend auf diesen Plattformen und Tools aufbauen:

n8n (oder gerne etwas Ähnliches) orchestriert den KI-Ansatz mit Workflow-Agenten, integriert interne Tools wie JIRA, unterstützt die Zusammenarbeit von mehreren Menschen (im Gegensatz etwa zu Claude Code), nutzt RAG. Ermöglicht Multiuser-Workflows und wird für „always-on"-Aufgaben verwendet (überwachen von Logs aus PROD). Damit können wir nicht nur neue Tool-Landschaft bauen, sondern ein hybrides Team von Menschen und KI Agenten.

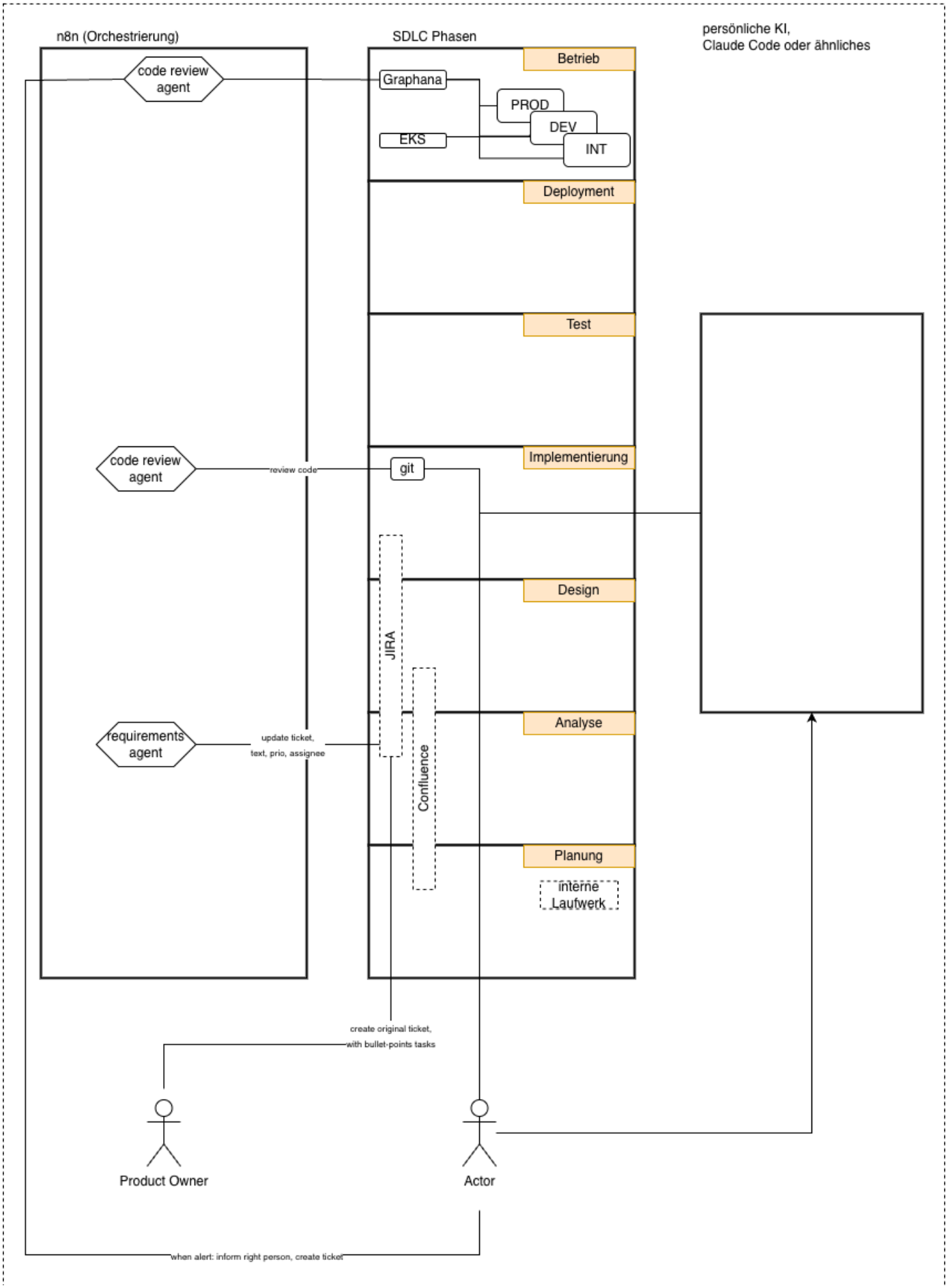
Claude Code (oder OpenCode oder Ähnliches) ist für den lokalen Einsatz gedacht: mit Refactoring, Codegenerierung inkl. Tests. Claude Code und n8n sollen sich gegenseitig möglicherweise ergänzen.

CI/CD-Build-Pipeline unterstützt die Phasen Implementierung, Test und Deployment und ist damit ein guter Kandidat für die Erweiterung mit KI:

- Code-Review von PRs mit CodeRabbit
- Log-Analyse für akute und potenzielle Probleme
- FinOps-Bots für die Berechnung von Kosten beim Deployment zu AWS
- Fraud-Detection
- Alerting der richtigen Personen bei Problemen …

Ich kann kaum aufhören, darüber nachzudenken, wo man das Potenzial von KI nutzen könnte. Wichtig ist, die Werkzeuge iterativ auszuprobieren, darüber zu reflektieren und sich nicht entmutigen zu lassen: Falls Framework A nicht gut funktioniert, kommt wahrscheinlich Framework B mit besseren Eigenschaften – und wir haben dafür in unserer Infrastruktur bereits Platz reserviert.

Landkarte: KI-gestützte SLDC



Ethische Aspekte und Verantwortung

Governance, Datenschutz, Compliance, Ethik. KI kann helfen, Richtlinien und Policies konsequenter durchzusetzen. Gleichzeitig entsteht eine neue Gefahr: Es ist verlockend und technisch einfach, sensible oder personenbezogene Daten in öffentliche LLMs hochzuladen – oft ohne böse Absicht, aber mit realen Konsequenzen.

Dieses Risiko wächst mit den Reifestufen. Spätestens auf Stufe 3 (unternehmensweite Workflows) und Stufe 4 (Agenten-Ökosystem) wird die Einhaltung von Governance-Regeln zur kritischen Voraussetzung – nicht zur Nachbedingung.

Klassische Schutzmaßnahmen wie Kryptographie und rollenbasierte Zugriffsrechte sind notwendig, aber vermutlich nicht ausreichend. Ein KI-Ökosystem braucht neue Konzepte: Alte Lösungen lassen sich eher nicht einfach übertragen.

Ich behandle dieses Kapitel bewusst als Platzhalter – nicht weil das Thema unwichtig ist, sondern weil ich es ehrlich adressieren will: Das Problem ist erkannt, die technischen Antworten sind noch in Entwicklung.