Sycophancy! Die fehlenden Produktmanager künstlicher Intelligenz

KI-Tools, die sofort mit dem Coden beginnen, sind nicht hilfreich — sie sind gefällig. Und genau diese Gefälligkeit führt in der Softwareentwicklung oft zu massiver Nacharbeit, Fehlentwicklungen und verlorener Zeit.

2026-04-02

(english below)

Deutsch

Die meisten AI-Coding-Tools haben ein grundlegendes Problem:

Sie beginnen sehr schnell mit der Lösung, bevor das eigentliche Problem sauber verstanden wurde.

Genau darin liegt eine der teuersten Schwächen moderner KI-Systeme: **Sycophancy** — *die Tendenz, auf einen Prompt möglichst überzeugend zu reagieren, statt zunächst zu prüfen, ob die zugrunde liegende Anforderung überhaupt schon klar genug ist.*

Im Kontext von Code-Generierung sieht das oft so aus:

Ein Tool bekommt eine Idee als Prompt und beginnt sofort, eine Anwendung daraus abzuleiten. Das Ergebnis wirkt auf den ersten Blick beeindruckend — häufig entstehen in kurzer Zeit klickbare Oberflächen, schlüssig formulierte Features und sauber aussehender Code. Aber genau an diesem Punkt beginnt oft das eigentliche Problem.

Denn Software scheitert in der Praxis selten zuerst am Schreiben von Code. **Sie scheitert daran, dass Anforderungen unvollständig, unscharf oder implizit bleiben.**

> *Welche Nutzerrollen gibt es?*

> *Welche Prozesse soll die App tatsächlich abbilden?*

> *Welche Ausnahmen und Sonderfälle gehören zur Geschäftslogik?*

> *Was ist zwingend notwendig, was nur „nice to have“?*

> *Und wie soll sich das Produkt in bestehende Systeme, Abläufe und Markenwelten einfügen?*

Wenn diese Fragen nicht sauber geklärt sind, produziert auch sehr gute KI vor allem eines: **plausible Missverständnisse.**

Genau hier setzt APA anders an.

Bevor APA eine einzige Zeile Code generiert, führt der APA Product Manager Agent einen strukturierten Anforderungsschritt durch. Er spricht nicht primär mit Entwicklerinnen oder Entwicklern, sondern mit der Person, die das fachliche Problem wirklich versteht.

Dabei geht es nicht zuerst um technische Implementierungsdetails, sondern um die eigentliche Produktlogik:

> *Was soll die Anwendung leisten?*

> *Wer arbeitet mit ihr?*

> *Welche Informationen, Rollen und Entscheidungen müssen abgebildet werden?*

> *…*

Logik klar, einfach, aber effektiv abzufragen ist vielleicht eine der am Meisten unterschätzten Qualitäten moderner KI-basierter Software-Generierung. Denn sie hat das Potenzial den häufig enormen Rework Aufwand signifikant zu reduzieren. 15 Minuten mehr die in von Product Owners in Anforderungsanalyse investiert werden, können schnell Wochen an Iterationsaufwand im Nachhinein einsparen.

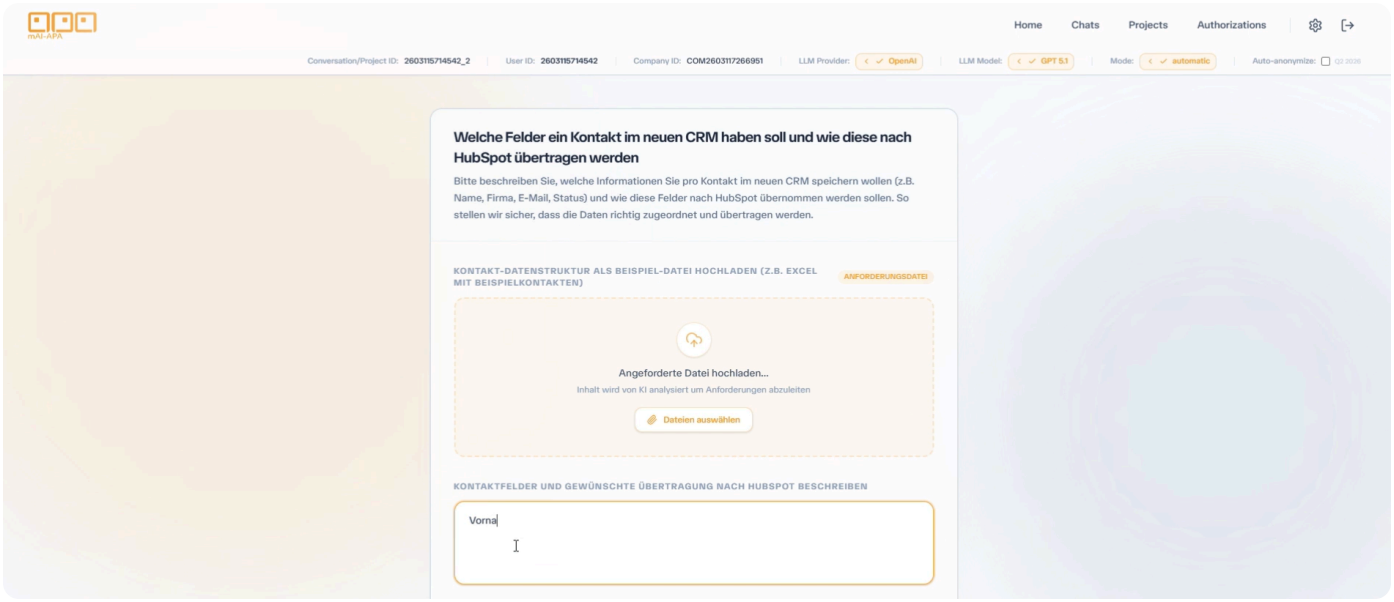

Abb: Screenshot des mai-apa.de Product-Managers

Der Product Manager ist ein spezialisiertes Feature der mai-apa Plattform was die Herausforderung *"Es ist einfach KI das Fragen beizubringen - aber es ist schwierig dass die richtigen Fragen, in der richtigen Tonalität gestellt werden"* betrachtet. Bei mAI APA werden die Personen der Fachabteilung geführt um wichtige Fragen konzeptioneller Art zu klären. Frage wie "Möchtest du eine Postgres Datenbankstruktur?" sind hier Fehlanzeige!
APA überlegt genau welche Informationen Nutzende vermutlich haben und fragt beispielsweise "Sollen verschiedene Personen diese Software nutzen können?". Die Übersetzung in technische Sprache (z.B. postgres ja oder nein) erfolgt dabei implizit und im Hintergrund.
Dazu können auch konkrete Materialien einfließen: I-Guidelines, Screenshots bestehender Tools, Referenzen und Beispieldaten. Aus diesen Informationen entsteht eine strukturierte Spezifikation — nicht nach mehreren Workshops und Übergaben, sondern in kurzer Zeit und mit der richtigen Person im Zentrum des Prozesses.

Warum ist das so entscheidend?

Weil der größte Qualitätshebel in KI-basierter Software-Generierung nicht nur im Modell oder Framework liegt, sondern im Input. Genauer: in der Qualität der Spezifikation, aus der das Modell arbeitet.

Wir haben APA-Outputs mit generischen AI-Outputs auf Basis desselben Ausgangsprompts verglichen. Der Unterschied lag nicht einfach in „besserem Code“, sondern in der Struktur des Ergebnisses: Die APA-Version war architektonisch konsistenter, näher an realen Geschäftsanforderungen und deutlich näher an einem produktionsreifen Zustand.

„Garbage in, garbage out“ gilt eben auch im Zeitalter generativer KI.

Genau für diesen Schritt haben wir APA gebaut.

Wenn wir Fachabteilungen vor der Entwicklung durch einen strukturierten PM-/Requirements-Prozess führen, verschieben wir Aufwand nach vorne — aber vermeiden teure Rework-Schleifen später, wo Änderungen um ein Vielfaches teurer sind .

Laut einer oft referenzierten Branchenlogik entstehen Fixkosten von Fehlern exponentiell später: Ein Anforderungsfehler, der erst im Systemtest korrigiert wird, kostet ungefähr 15x so viel wie ein früher Fix

Für die betriebswirtschaftliche Betrachtung reicht eine konservative Rework-Annahme. Operativ ist das Risiko bei ungeklärten Anforderungen und direkter KI-Generierung aber häufig höher als in klassischen Projekten, weil größere Teile des Outputs später angepasst, ersetzt oder ganz verworfen werden müssen.

> *Hashtags: #KI #AI #ProductManagement #Requirements #SoftwareEntwicklung #BuildInPublic #APA*

English

Most AI coding tools face a fundamental problem:

They jump into the solution far too quickly, before the actual problem is clearly understood.

This points to one of the most costly weaknesses of modern AI systems: **Sycophancy** — *the tendency to respond to a prompt as convincingly as possible, rather than first checking whether the underlying requirement is even clear enough.*

In the context of code generation, it often looks like this:

A tool receives an idea as a prompt and immediately begins deriving an application from it. At first glance, the result looks impressive — clickable interfaces, coherently formulated features, and clean-looking code often emerge in a very short time. But this is exactly where the real trouble begins.

In practice, software rarely fails because of the code-writing process itself. **It fails because requirements remain incomplete, fuzzy, or implicit.**

> *What are the user roles?*

> *Which processes is the app actually supposed to map?*

> *Which exceptions and edge cases belong to the business logic?*

> *What is mandatory, and what is just "nice to have"?*

> *And how should the product fit into existing systems, workflows, and brand identities?*

If these questions aren't properly clarified, even excellent AI produces primarily one thing: **plausible misunderstandings.**

This is exactly where APA takes a different approach.

Before APA generates a single line of code, the **APA Product Manager Agent** carries out a structured requirements phase. It doesn't primarily talk to developers, but to the person who truly understands the domain-specific problem.

The focus here isn't on technical implementation details, but on the actual product logic:

> *What should the application achieve?*

> *Who will be working with it?*

> *Which information, roles, and decisions need to be mapped?*

> *...*

The ability to test logic in a clear, simple, yet effective way is perhaps one of the most underrated qualities of modern AI-based software generation. This is because it has the potential to significantly reduce the often enormous amount of rework

required. An extra 15 minutes invested by product owners in requirements analysis can quickly save weeks of iteration effort down the line.

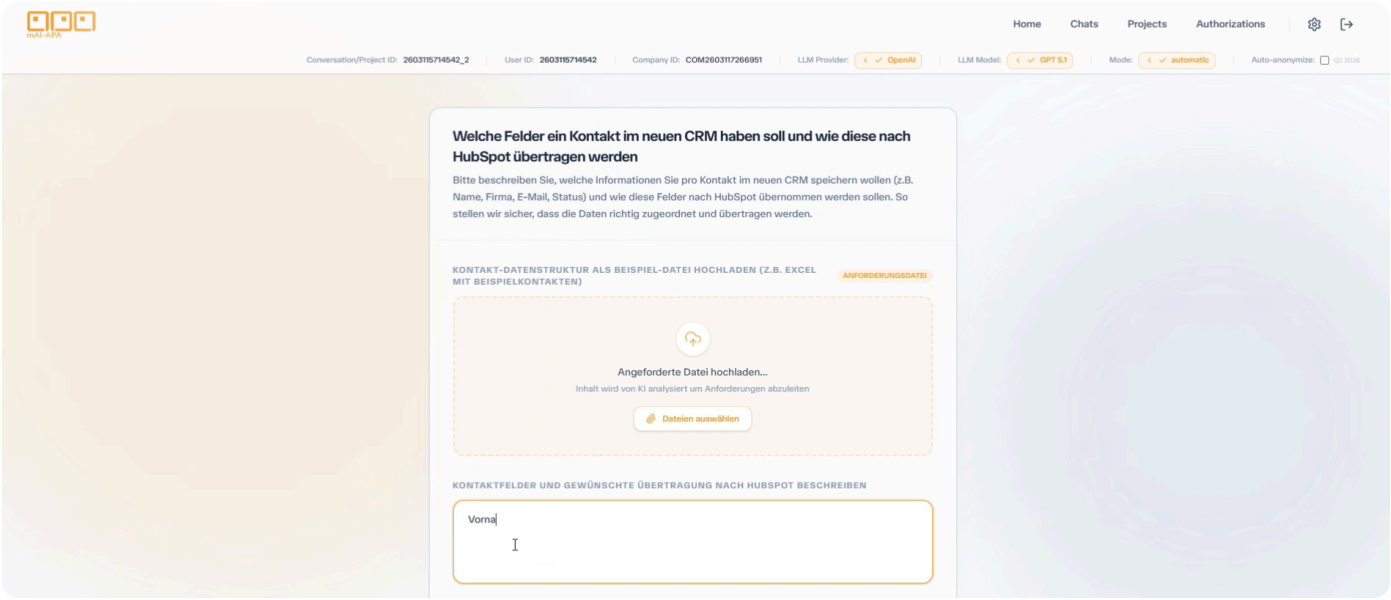

Abb: Screenshot of the mai-apa Product Manager

The Product Manager is a specialized feature of the mAI-APA platform that addresses the challenge: *"It's easy to teach AI to ask questions — but it's difficult to ensure the right questions are asked in the right tone."* At mAI APA, experts from the respective business departments are guided through clarifying essential conceptual questions.

You won't find questions like "Do you want a Postgres database structure?" here!
Instead, APA carefully considers what information users likely have and asks, for example: "Should different people be able to use this software?" The translation into technical language (e.g., Postgres yes or no) happens implicitly in the background.

Specific materials can also be integrated: Brand guidelines, screenshots of existing tools, references, and sample data. A structured specification emerges from this information — not after multiple workshops and handovers, but in a short time, with the right person at the center of the process.

Why is this so crucial?

Because the biggest quality lever in AI-based software generation lies not just in the model or framework, but in the input. More precisely: in the quality of the specification the model

works from.

We have compared APA outputs with generic AI outputs based on the same initial prompt. The difference wasn't simply "better code," but the structure of the result: the APA version was architecturally more consistent, closer to real business requirements, and significantly closer to being production-ready.

"Garbage in, garbage out" still applies in the age of generative AI.

We built APA specifically for this step.

When we guide business departments through a structured PM/requirements process before development, we shift the effort to the early stages—but avoid costly rework cycles later on, when changes are many times more expensive.

According to a frequently cited industry principle, the fixed costs of errors increase exponentially over time: A requirements error that is not corrected until system testing costs approximately 15 times as much as an early fix

From a business perspective, a conservative rework estimate is sufficient. Operationally, however, the risk associated with unresolved requirements and direct AI generation is often higher than in traditional projects, because larger portions of the output must later be adapted, replaced, or discarded entirely.

> *Hashtags: #AI #ProductManagement #Requirements #SoftwareDevelopment #BuildInPublic #APA*