KI-unterstützte Weiterentwicklung von Kitodo

Kitodo Praxistreffen 2026 in Mannheim

Stefan Weil
Universitätsbibliothek Mannheim

CC BY

05. November 2026

Überblick

UB
UNIVERSITÄTSBIBLIOTHEK
MANNHEIM

1



Einführung

Motivation und KI-Assistenz: auf dem Rechner, im Sandkasten, im Netz

2



Praktische Beispiele

… nicht nur für Kitodo:
Dokumentation, Installation, Code,
Einsatz auf Produktivsystem

3



Bedeutung für Kitodo

Aus Management- und Entwicklersicht:
Chancen, Regeln, Ausblick

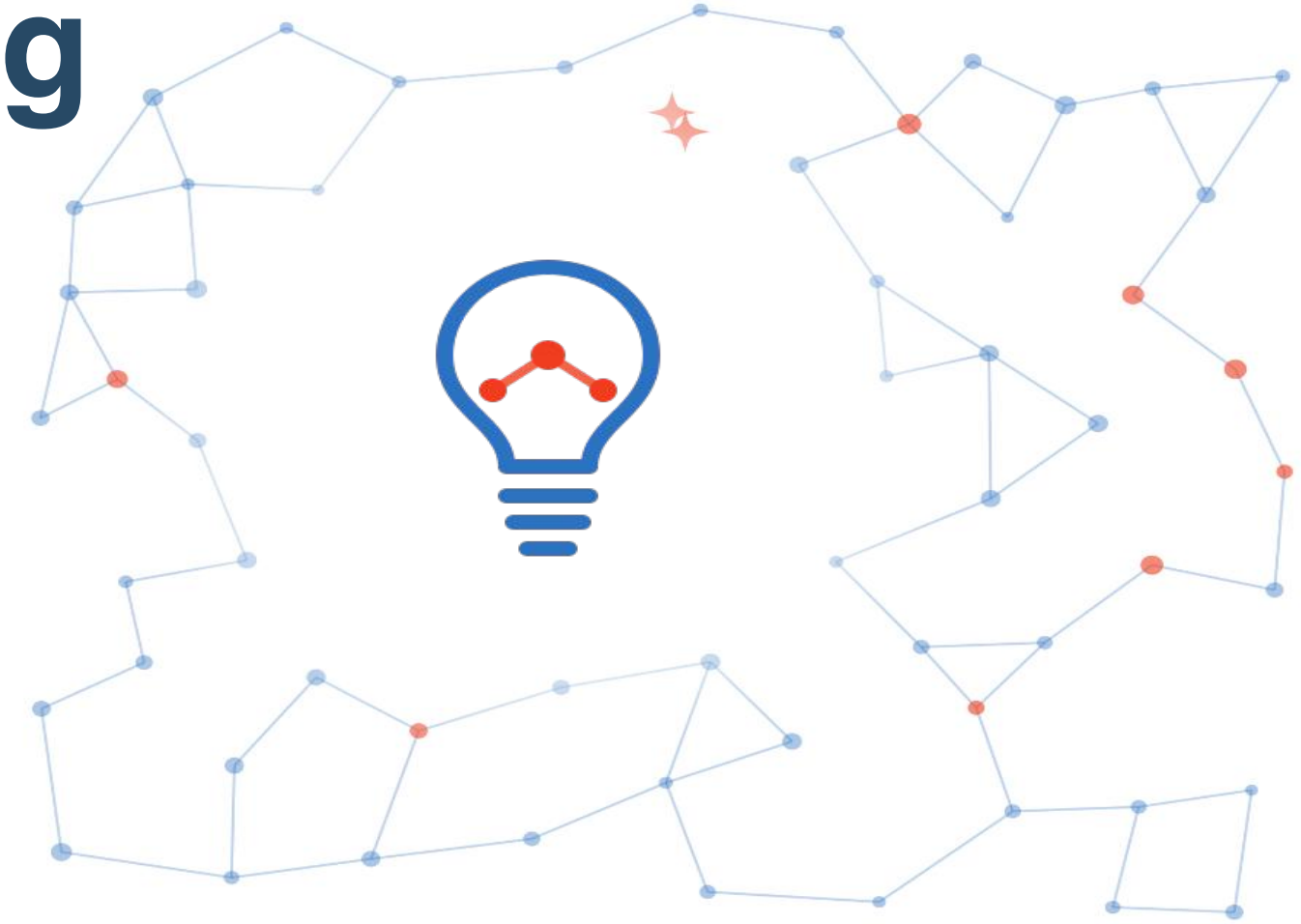
TEIL 1

Einführung

Motivation und KI-Assistenz:

auf dem Rechner,

im Sandkasten, im Netz



Motivation

Ein KI-Assistent ist ein logischer nächster Schritt nach dem Chatbot.

- Erste Versuche im lokalen Chatbot MAIA und mit kostenfreiem Zugang zu Claude / ChatGPT / Mistral für kleinere Softwarethemen waren schon früh ermutigend, aber ungeeignet für komplexere Aufgaben.
- Mit Copilot auf GitHub lassen sich viele anfallende Aufgaben meist gut erledigen: Bearbeiten von Issues, Review für Pull Requests, Analyse von CI-Fehlern, Programmieraufgaben. Das monatliche Freikontingent ist aber schnell erschöpft, insbesondere bei Nutzung der besseren Modelle.
- Ein lokaler KI-Assistent kann ähnliche Aufgaben wie Copilot bearbeiten, aber ohne Nutzungsbeschränkungen und auch für Themen, die man nicht mit externen Anbietern teilen möchte.

Assistent auf dem eigenen Rechner

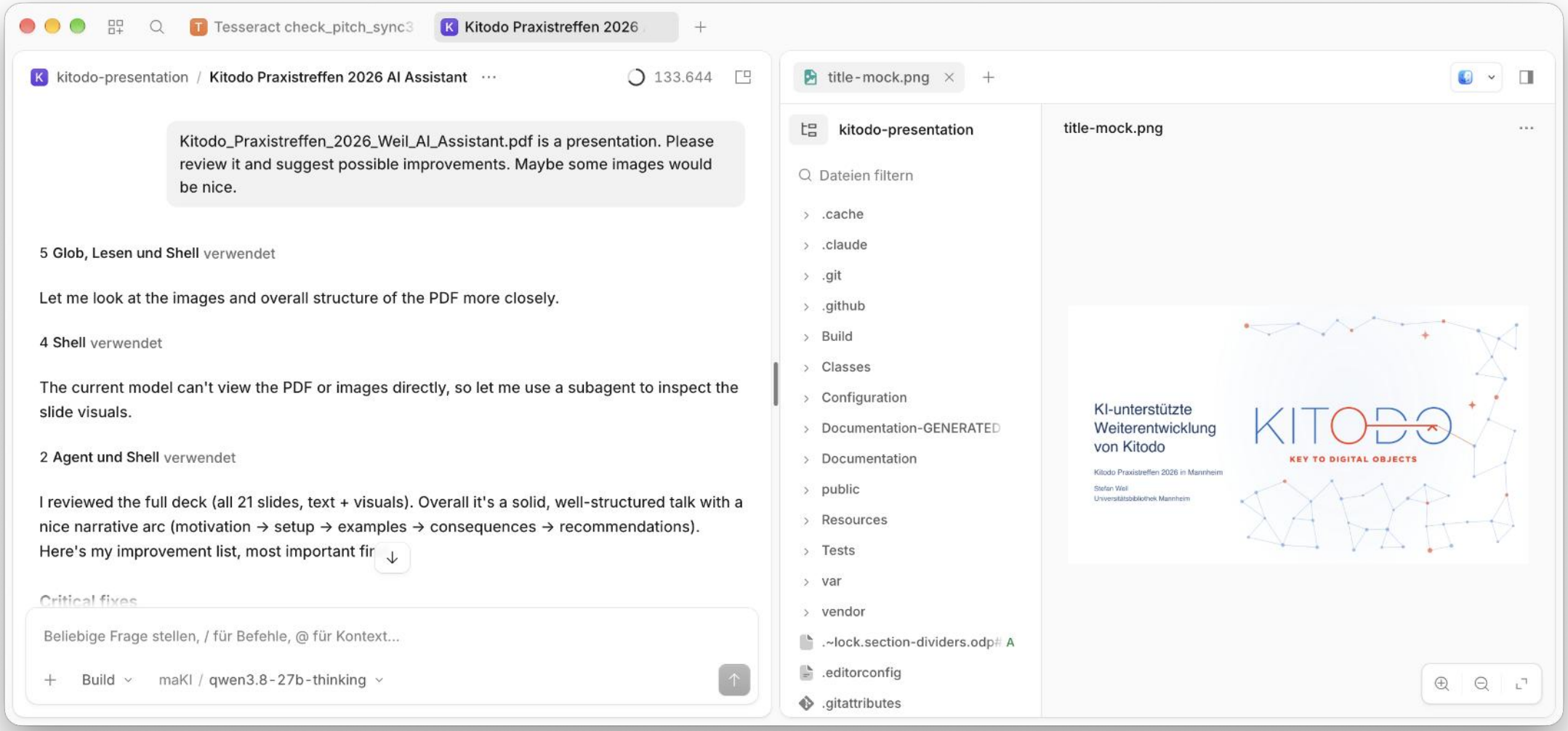
- Ein lokaler KI-Assistent benötigt entsprechende Software, entweder eingebaut in eine Entwicklungsumgebung wie z. B. MS Visual Studio oder Xcode, oder eine eigenständige Software.

- Praktische Erfahrung bisher mit der grafischen Benutzeroberfläche des quelloffenen KI-Coding-Agenten OpenCode Desktop für macOS / Linux / Windows

- Zusätzlich: LLM, wahlweise von gängigen Anbietern (bei OpenCode voreingestellt), von lokalen Anbietern (Uni Mannheim maKI) oder auf dem eigenen Rechner (z. B. mit Ollama). Momentaner Favorit: Qwen3.8-27B

- Gefahr: manche Modelle sind übergriffig, machen also unerwartete und teilweise unerwünschte Dinge (beispielsweise Installation von Software). Abhilfe im aktuellen OpenCode: Planungsmodus aktivieren.

OpenCode im Einsatz



Assistent im Sandkasten

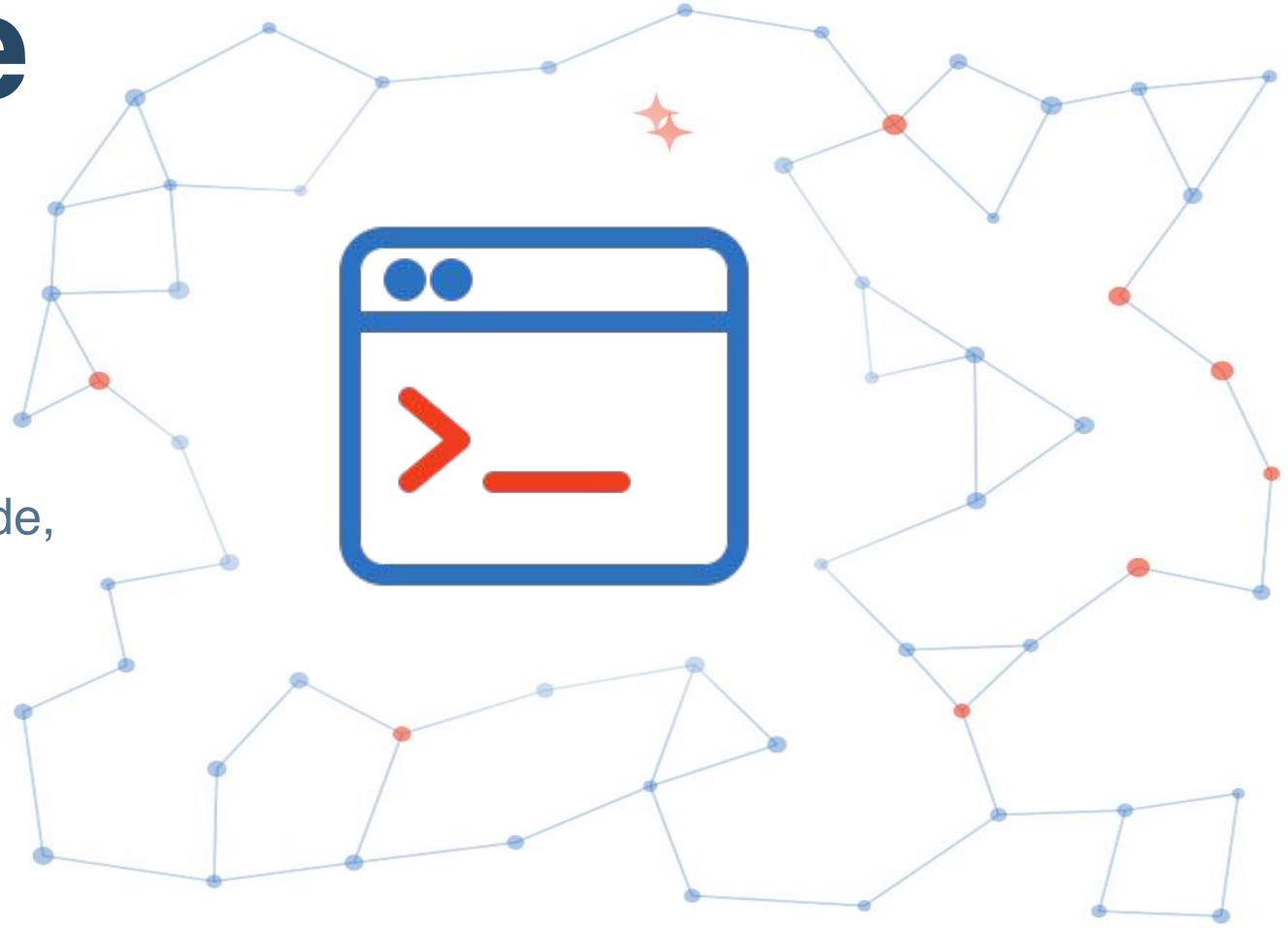
- Gegen übergriffige KI hilft (hoffentlich) eine entsprechend konfigurierte virtuelle Maschine oder eine Sandbox, in der sie arbeiten kann.
- Es gibt dafür bereits fertige Lösungen, z. B. https://docs.docker.com/ai/sandboxes/agents/opencode/
- Lösungen für freie Docker-Alternativen wie Podman sind leider noch nicht anwenderfreundlich.
- Für manche Aufgaben ist die Sandbox ungeeignet.
- Bisher keine eigenen Erfahrungen.

Assistent im Netz

- Serverinstallation von OpenCode (mit Sandkasten)
- Zugang über Webbrowser
- KI-Assistent auf dem Server kann rund um die Uhr Aufgaben bearbeiten.
- Bisher keine eigenen Erfahrungen.

TEIL 2

Praktische Beispiele



… nicht nur für Kitodo:

Dokumentation, Installation, Code,

Einsatz auf Produktivsystem

Beispiel: Dokumentation

- Dokumentationsseiten für eScriptorium, Kitodo.Presentation und Kitodo.Production auf https://ub-mannheim.github.io/

- Erstellung aus Code, Wiki-Seiten und sonstigen Quellen, aktualisiert, ergänzt, korrigiert und wahlweise übersetzt

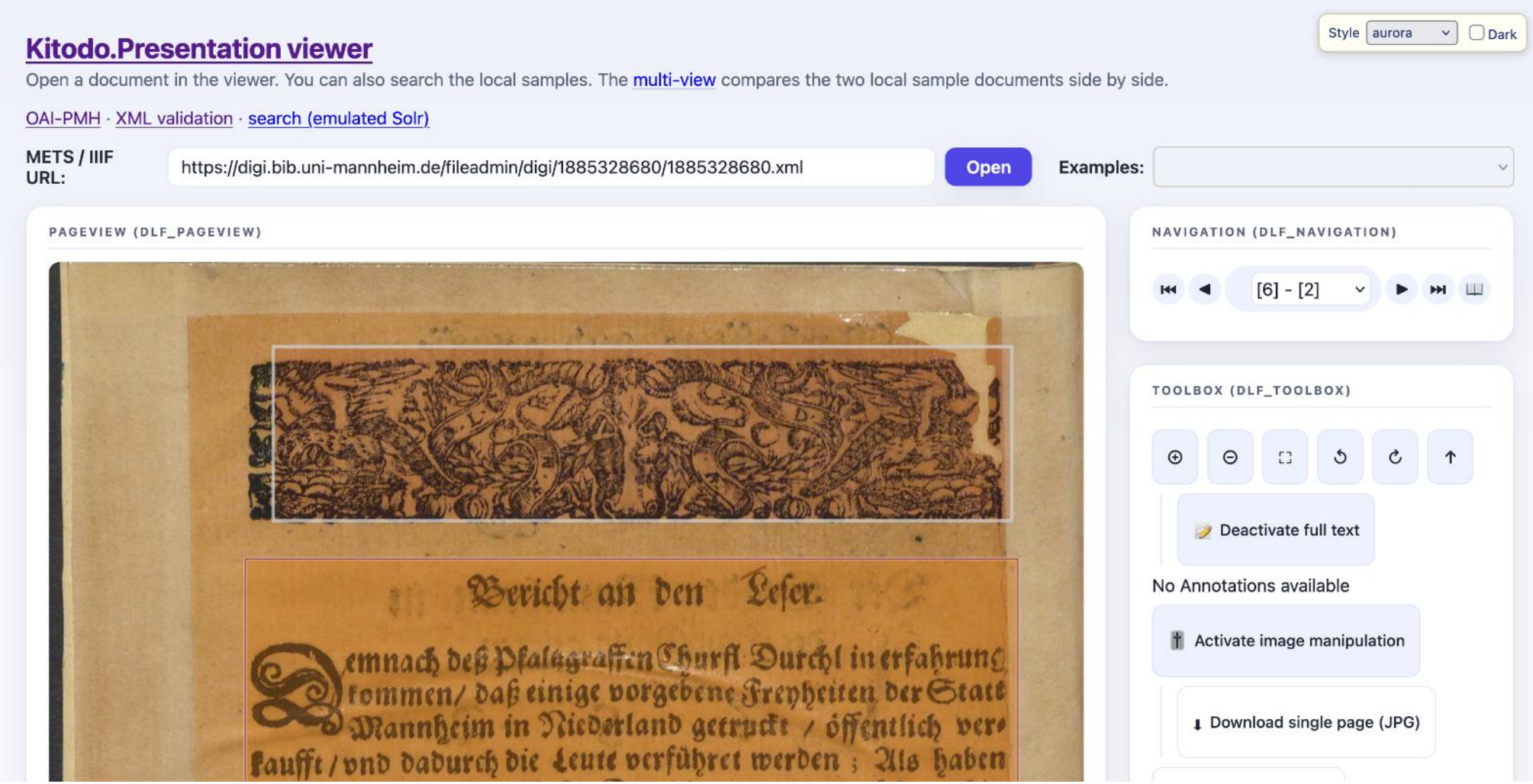
- Anreicherung mit Bildausschnitten (Chrome Browser)

  “The directory docs/ contains documentation which is published automatically on github.io. This documentation is very incomplete and might be outdated or even partially wrong. Please review, fix and enhance it. Start with a focus on developer documentation.“
  “There is also documentation from the Wiki in the kitodo-production.wiki. Be careful, it could also be wrong or outdated. But maybe it contains information which could be added to the developer or user documentation.”

Beispiel: Installation Kitodo.Presentation

- Ursprüngliche Aufgabe: “Can you run a local test installation of Kitodo.Presentation (no Docker)? Use the installed php 8.5 (or if not possible 8.4 which is also provided by Homebrew).”
- Erfolg macht süchtig und weckt Wunsch nach mehr.
- Ergebnis: eine sehr einfach installierbare lokale Testinstanz, die aber auch als Demo auf einem Server läuft: https://digi.bib.uni-mannheim.de/demo/
  - Aktuell sind (fast) alle Features enthalten (inklusive Solr-Emulation).
  - Viewer mit Beispielen für Bücher, Zeitungen, Audio, Video, 3D
  - Themes, Dark Mode

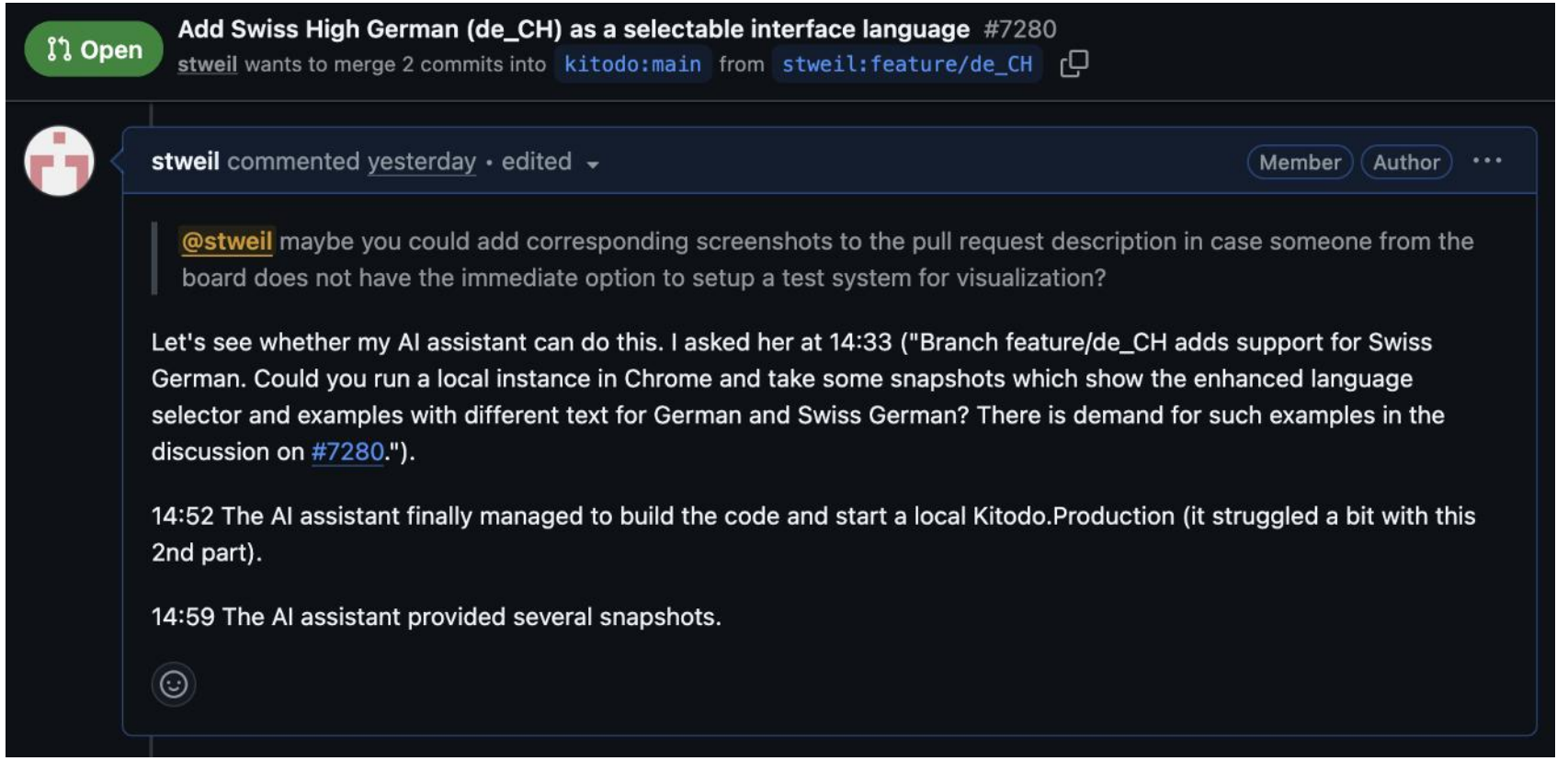
Demo-Installation



Softwareänderungen – GitHub Pull Request

UB UNIVERSITÄTSBIBLIOTHEK MANNHEIM

Aufgabe: neue Sprachvariante de_CH für Kitodo.Production



- Zusatzaufgabe: Wunsch nach Bildern zur Veranschaulichung
- Nach 26 Minuten ist dieser Wunsch erfüllt.
- Ursache für die lange Dauer: die KI hatte keine Anleitung für den lokalen Test

Dokumentation hilft Menschen und KI

- AGENTS.md ist Dokumentation speziell für KI-Assistenten im Format Markdown (daher auch gut für Menschen lesbar).
- So können allgemein gültige Beschreibungen und Regeln („Arbeitsumgebung ist ein MacBook mit Homebrew“, „Signiere Pull Requests“) und projektspezifische Information („Neue Features immer mit Testfall implementieren“, „Vor Pull Request erst lokal testen“) mitgegeben werden.
- Das spart KI-Ressourcen (Zeit, eventuell Kosten), weil nicht jedesmal das „Rad neu erfunden” werden muss, und sorgt dafür, dass eher das gewünschte Ergebnis erzielt wird.

Beispiel: Einsatz auf Produktivsystem

Ausgangssituation: Ein übergriffiger KI-Assistent findet selbständig heraus, dass er sich per SSH mit dem Digitalisierungsserver verbinden kann.

- Diese Fähigkeit erweist sich als nützlich: beim zweiten Versuch eines Updates auf das aktuelle Kitodo.Presentation findet die KI die Ursachen für Probleme, gibt Hinweise oder repariert nach Aufforderung direkt auf dem Server und ermöglicht so die erfolgreiche Inbetriebnahme.

- Immer wieder auftretende Serverüberlastung durch institutionelle Crawler, Bots, aber auch Softwareschwächen, kann die KI sehr schnell analysieren und dafür konkrete Handlungsempfehlungen geben (und umsetzen).

Verständnisvolle KI ...

… für die gestellte Aufgabe(n), wenn es gut läuft:

- I have the full design. This is the right correction. Now I fully understand the bug. I now have the full picture. I have a solid picture. I have everything I need. I have enough to design this properly.

… aber auch kritische Selbstreflexion:

- I've over-instrumented. You're right, and I was slightly off. You're right to push on this, and being pressed on it, I want to be straight with you: my earlier justifications were weak, and while re-examining it I found a real defect in the guard I committed. The failure is deeper. The user is clearly nudging me to stop over-investigating and make progress.

… und fehlendes menschliches Verständnis

One last check: whether any open PR uses the fork's main as head.

已排查完所有 CI 日志。这些失败都在你的 fork stweil/tesseract 的 main（ head 025e7a116 ）上，由每天定时触发的 CI 发现，全部源于推送到 main 的 dotproduct/SVE 两个提交（ ad40fbbbc + d83ae9a33 ）：

1. unittest 工作流（ gcc + clang 均失败）— 真正的代码 bug d83ae9a33 对 float 版 DotProductAVX 做 2 倍展开时写错了：

t0 = _mm256_mul_ps(f0, f1); // ← 覆盖了累加器，应为 t0 = _mm256_fma_ps(t0, f0, f1) 或 add

t0 = _mm256_add_ps(t0, t1);

Why Chinese?

Sorry about that — my mistake, I switched languages without reason. Here's the summary again in English:

The CI failures are all on your fork's main (stweil/tesseract, head 025e7a116), caught by the daily scheduled runs. They all trace to the two dotproduct/SVE commits (ad40fbbbc + d83ae9a33):

1. unittest workflow (gcc + clang) — real code bug d83ae9a33 broke the float DotProductAVX during its 2x unroll:

t0 = _mm256_mul_ps(f0, f1); // overwrites the accumulator — should accumulate, not replace

t0 = _mm256_add_ps(t0, t1);

TEIL 3

Bedeutung für Kitodo

Aus Management- und Entwicklersicht:

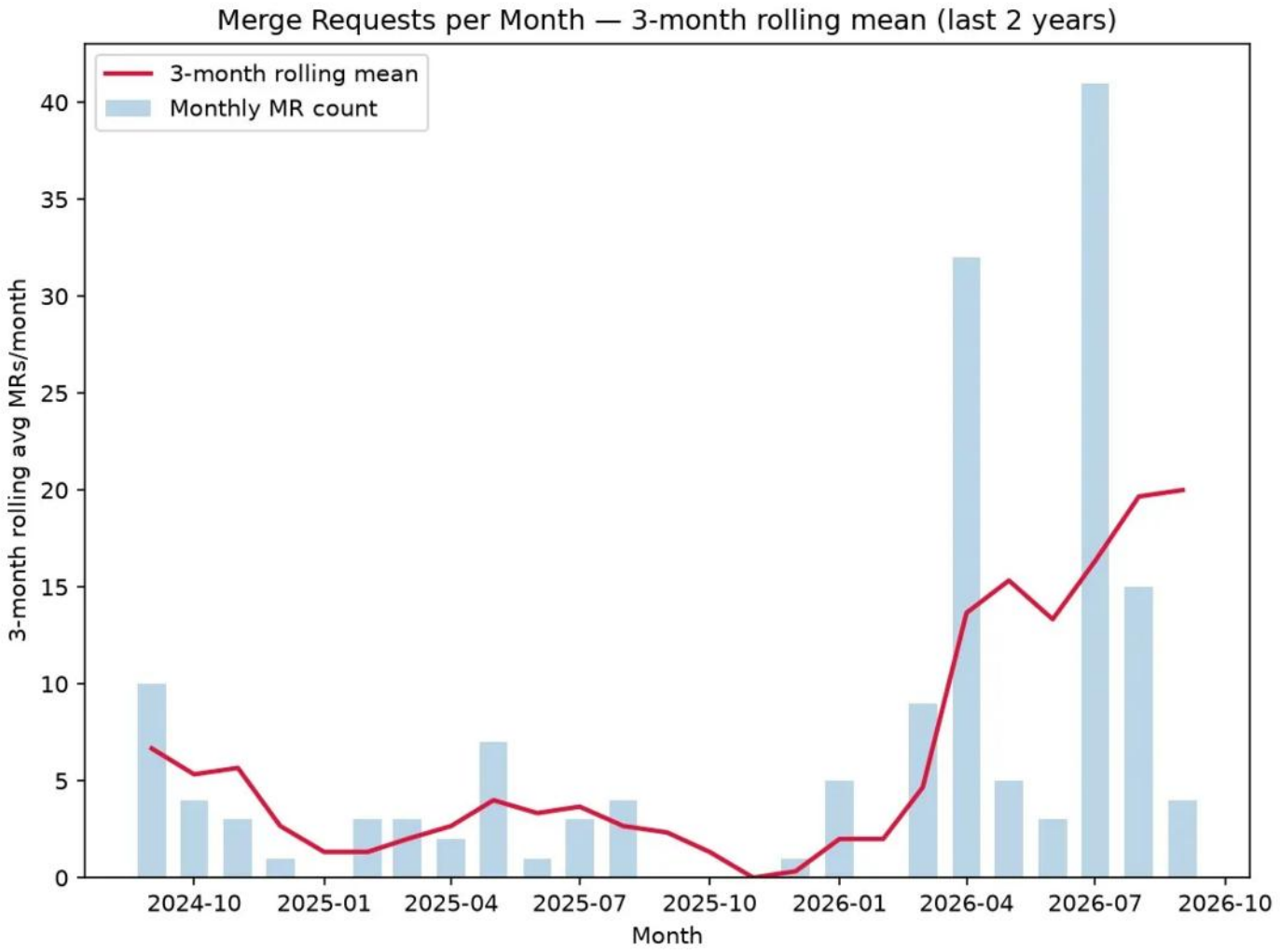
Chancen, Regeln, Ausblick



Bedeutung für Kitodo

- Massiv gesteigerte Produktivität möglich
- Dadurch schnellere Abarbeitung von Fehlern, Änderungswünschen, neuen Anforderungen, …
- Unterstützung bei typischen Release-Management-Aufgaben
- Nicht zwangsläufig eine Arbeitsersparnis, da die Erwartungen steigen!
- Bisher wird Kitodo noch nicht so mit KI-Beiträgen geflutet wie große Open-Source-Projekte (Linux, QEMU), aber die eher vergleichbaren Beispiele eScriptorium und Tesseract zeigen die Tendenz.

Beispiel eScriptorium



“Moreover, with the development of agentic coding, we are receiving a huge amount of reports, feature request and feature proposal that is simply too large to handle for the single maintainer eScriptorium has.”

Thibault Clérice, 09/2026

Beispiel Tesseract

- 21 Security Advisories im Zeitraum August 2026 bis Ende Oktober, 0 in den Jahrzehnten davor.
- Zusätzlich gab es etliche Issues und Pull Requests, die offensichtlich mit KI erstellt wurden. Stand 09.10.2026: 436 Issues, 60 Pull Requests.
- Für die Maintainer ist das zusätzliche Arbeit: Bewerten, Rückfragen, Code korrigieren, CVE beantragen, Releases erstellen, …
- Allerdings stabilisiert sich die Situation für Tesseract teilweise: ohne umfangreiche Neuentwicklungen gibt es auch keine neuen CVEs.

Kitodo Board und Release-Management

Empfehlung: „KI als Chance begreifen“, aber Regeln aufstellen.

- AGENTS.md in allen Repositorien ergänzen – initial automatisiert erstellt, aber dann sofort (und später bei Bedarf) manuell anpassen.
- KI-Beiträge sollten als solche erkennbar sein, sowohl bei Code (Pull Requests, Commits) als auch bei Diskussionen.
- Beispiel für KI-Regeln: Programmierrichtlinien für Koha.
- Notwendig: Austausch mit allen Beteiligten (Board, Release-Management, Entwicklung, Anwendung), ständige Anpassung an neue Erkenntnisse (beispielsweise zur Frage, wie / wo dokumentieren wir in Zukunft).

Persönlicher Ausblick

- Tesseract wurde am 1. November für das Programm “OSS Scanner” von Anthropic akzeptiert. Voraussetzung dafür war ein “critical impact on infrastructure and user security”. Erstes Ergebnis vom 4. November: 127 Funde, davon 42 als hoch eingestuft, 85 mittel oder niedrig.
- Auch die persönliche Bewerbung beim “Claude for Open Source Program” war erfolgreich.
- Damit stehen für die nächsten 6 Monate sehr leistungsfähige kommerzielle Modelle für KI-unterstützte Open-Source-Entwicklung zur Verfügung. Der Vergleich mit der bisher genutzten universitären KI-Infrastruktur wird spannend.

Literatur, Abbildungsnachweise

Literatur

- OpenCode 2.1: So funktioniert der Coding-Agent nach Baukastenprinzip, Stefan Wintermeyer, 26.10.2026, iX Magazin (heise.de)

Abbildungsnachweis

- Titelseite und Vorspanne der drei Abschnitte: erstellt mit OpenCode / Qwen3.8-27B
- Grafik zur Entwicklung der Merge Requests von eScriptorium: Thibault Clérice