ABAP & KI

Was heute wirklich geht – von der Chat-Anfrage zum Agenten

Webinar, 25.9.2026, 10:00 Uhr

supported by Cadaxo & Brandeis

Johann Fößleitner

Senior Consultant at Cadaxo

eMail: johann.foessleitner@cadaxo.com
LinkedIn: https://www.linkedin.com/in/foessleitnerj/

Beratungsschwerpunkte

- KI-gestützte ABAP-Entwicklung
- Clean Core & Clean Code
- ABAP Cloud, RAP & Fiori Elements
- Qualität, Performance & Output Management (dox42)

- **SAP Champion**

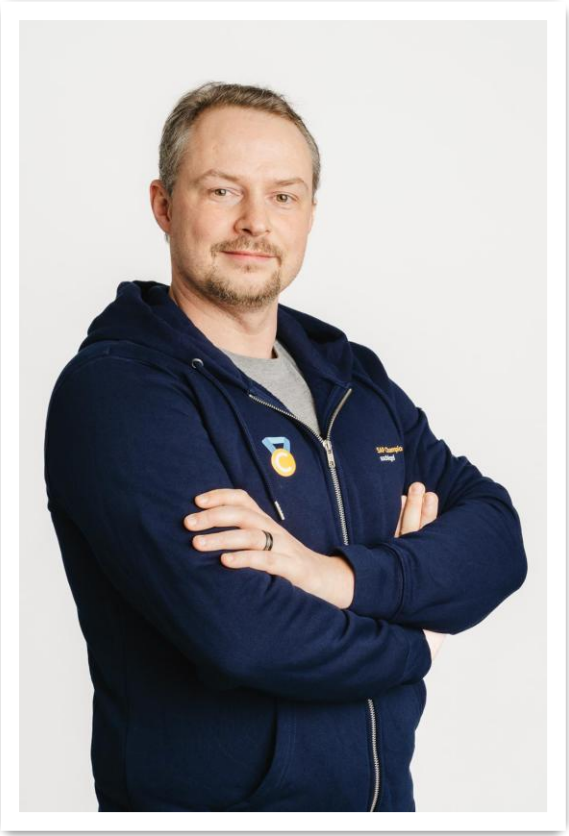
Domi Bigl

Senior Consultant at Cadaxo

eMail: dominik.bigl@cadaxo.com
LinkedIn: https://www.linkedin.com/in/domi-bigl

Beratungsschwerpunkte

- ABAP Development mit KI
- ABAP Cloud, RAP und S/4HANA Entwicklung
- Fiori Elements, SAPUI5
- Performanceoptimierung



Sören Schlegel

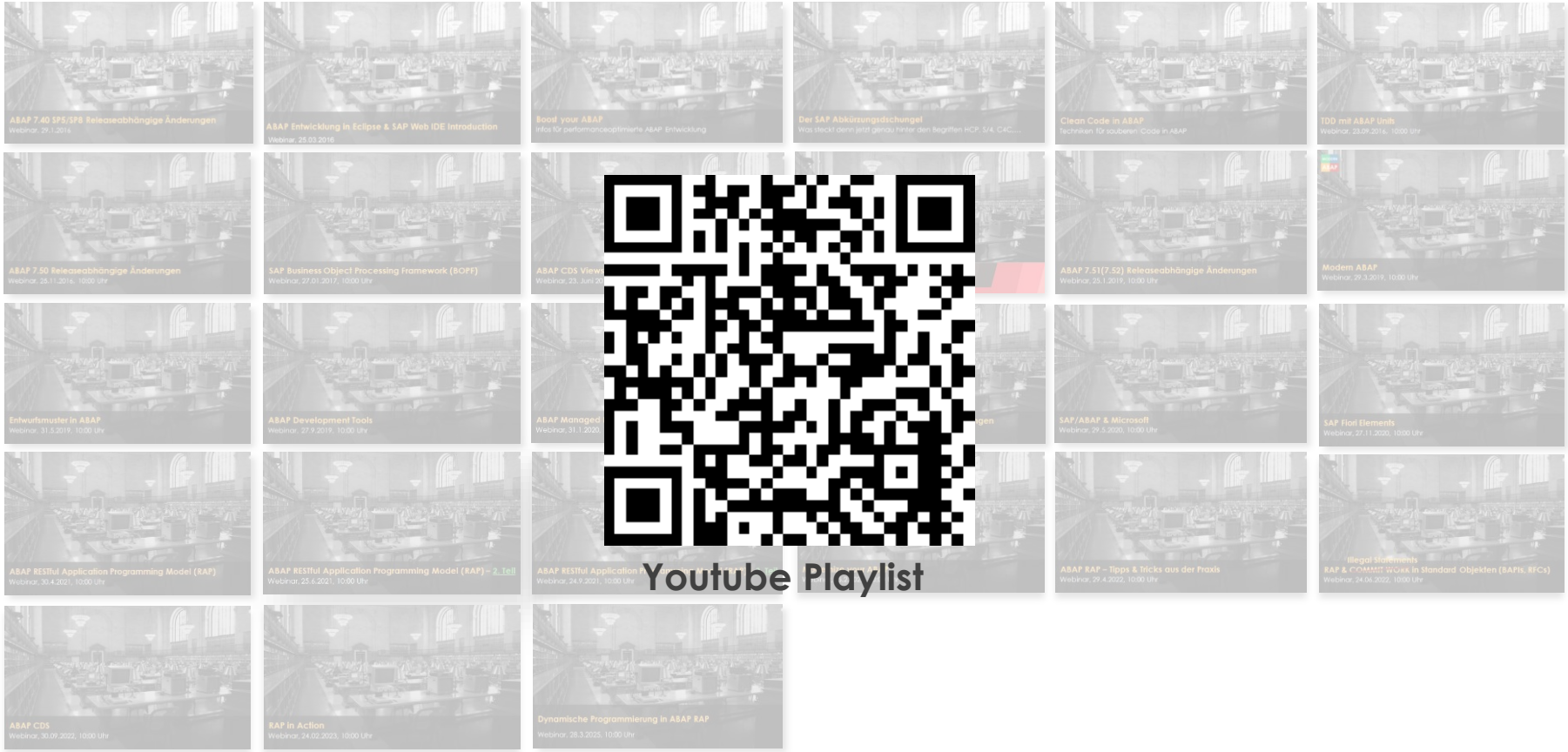
SAP Development Architect

Twitter: @SoSchlegel87
LinkedIn: https://www.linkedin.com/in/s%C3%B6ren-schlegel-55171bb4/

Themenschwerpunkte

- Konzeption & Management von Entwicklungsprojekten
- SAP Architektur moderner Anwendungen
- ABAP RAP und S/4HANA Entwicklung
- SAP CDS, OData und Cloud

- **SAP Champion**



Youtube Playlist

Unsere vergangenen Webinare

cadaxo

SAP® DEVELOPMENT MADE SIMPLE

- **Consulting, Workshops, Coaching, Produkte**
- ABAP, CDS, RAP, S/4 Extension, Code Migration, …
- https://www.cadaxo.com

Brandeis Consulting

- **Consulting, Schulungen, Workshops**
- ABAP, CDS, RAP, SQLScript, …
- https://www.brandeis.de

Community News

Rückblick: ABAPConf 2026

23. Juni 2026 – 5 Jahre ABAPConf – online und vor Ort in Mannheim

5
Jahre ABAPConf

2
Tracks live gestreamt

268
Peak online in Track 1

7.500
Views der Livestreams

Spendenaktion Deutsche Krebshilfe
Ziel 500 € – deutlich übertroffen: 2200 € - Danke an alle!

Nachschauen

- https://www.youtube.com/@abapconf
- https://www.itsfullofstars.de/2026/07/abapconf-sessions

Ausblick: Was jetzt ansteht

Die Events der nächsten Wochen für die SAP Developer

21.9. – 18.10.

Devtoberfest

Jeden Montag ABAP Day.
Schwerpunkte: Agentic AI, ABAP Cloud

Marcel, Andre, Olga, …

10.10.

SAP Inside Track München

Garching - Aenda folgt.
Anmeldung bereits möglich.

27. – 29.10.

SAP TechEd

Berlin - Keynotes, Expert Sessions und Developer-Content – auch teilw. on demand.

Ausblick: Was jetzt ansteht

Die Events der nächsten Wochen für die SAP Developer

13.11.

SAP CodeJam RAP Special Edition

Köln - SAP CodeJam für erfahrene RAP-Entwickler mit den neuesten Features, inklusive VS Code, MCP und agentischer Entwicklung.

Andre

17. – 18.11.

DSAG Zukunfststag Development & AK-Treffen

Frankenthal – TechEd-News zu KI und Agenten, aus DSAG-Sicht

Okt. / Nov. / Dez.

Workshop ABAP & KI

Brandeis in Mannheim
Cadaxo in Wien

Hörtipps

Drei Podcasts: KI-Überblick, SAP Output Management und digitales Lernen

Der KI-Podcast

ARD

KI verständlich erklärt, jede Woche neu

Die wichtigsten Entwicklungen rund um KI, unaufgeregt eingeordnet. Ideal für den Blick über den SAP-Tellerrand.

Spotify

We love Output

SAP Output Management
Frank Engert, Softway

Sommerreihe mit Interviews

Der Podcast rund um Formulare, Druck und Dokumentenausgabe in SAP.

Zu Gast u. a. Karl Kessler, Sören

Spotify

Education NewsCast

Thomas Jenewein, SAP

Digitales Lernen, Change Management und KI

Im EducationNewsCast dreht sich alles um digitales Lernen, künstliche Intelligenz und AI Adoption.

Spotify

ABAP & KI

KI und ABAP: das große Bild

Heute sehen wir uns die Entwickler-Seite an

heute im Fokus

KI für Entwickler

Die KI hilft mir beim Entwickeln

Werkzeuge
GitHub Copilot, Claude Code, Joule for Developers, Amazon Q Developer

Merke
Das Prinzip ist gleich, nur das Tool wechselt

Wird noch selten benötigt

KI in Anwendungen

Mein ABAP-Code ruft ein LLM auf

Werkzeug
ABAP AI SDK, Custom Development, Modelle über den Generative AI Hub

Beispiele
Freitexte zusammenfassen, Dokumente klassifizieren

Anderer Use Case

KI für Migration

Die KI hilft auf dem Weg zu S/4 und Clean Core

Beispiele
Alte Z-Programme verstehen, Prüfbefunde erklären und beheben

Werkzeuge
Joule oder dieselben Tools wie links

Unsere KI Landkarte

Landkarte

1	**Modell**	Der Motor. Austauschbar.	GPT · Claude · Gemini · Mistral · Llama
2	**Zugang**	Gleiches Modell, andere Hülle.	Browser · IDE-Plugin · CLI · API
3	**Kontext**	Was das Modell sieht.	Prompt · Dateien · AGENTS.md · Skills
4	**Zugriff**	Was sieht das Tool, was darf es tun.	MCP-Server
5	**Autonomie**	Wer führt.	Autocomplete → Chat → Agent

Heute live: Chat → Kontext → Zugriff → Autonomie

Das Modell

Der Motor. Austauschbar – und für sich alleine harmlos.

1. **Text rein, Text raus**

   Dein Prompt -> Modell -> Antwort

2. **Das Modell erzeugt Wahrscheinlichkeiten**

   Es sagt das nächste Wort vorher. Vereinfacht: Token für Token.

3. **Das Modell allein hat keinen Zugriff auf**

   SAP-Systeme, Dateien, Datenbanken, eure Unternehmensdaten

Wer baut die Motoren

GPT	OpenAI
Claude	Anthropic
Gemini	Google
Mistral	Mistral AI
Llama	Meta

Mistral und Llama sind auch selbst hostbar — deshalb tauchen sie in EU-Datenschutzdiskussionen auf.

! *Das Modell allein kann nichts als reden.* ***Und genau deshalb brauchen wir die anderen Ebenen unserer Landkarte.***

Der Zugang

Gleiches Modell, andere Hülle.

Browser / App

ChatGPT, Claude im Chatfenster

Copy-Paste. Kein Setup.

IDE-Plugin

GitHub Copilot in VS Code oder Eclipse

Sieht die offene Datei.

CLI

Claude Code im Terminal

Arbeitet im Workspace.

API

Eigener Aufruf aus Code

Für Integration ins Produkt.

Was oft verwechselt wird

GitHub Copilot ist kein Modell, sondern ein Tool für IDE-Integration — läuft mit GPT, Claude, …

Cursor ist ein eigener Editor, **Claude Code** ein CLI-Werkzeug. Beide sind Ebene 2, nicht Ebene 1.

Warum ABAP anders ist?

Trainingsdaten

Ein Modell kann gut, was millionenfach öffentlich vorliegt.

	Öffentlicher Code	KI-Unterstützung
JS / TypeScript, Python, Java	riesig	**stark**
CAP (Node / TypeScript)	groß	**sehr gut**
UI5 / Fiori Elements	ordentlich	**gut — Annotations prüfen**
Klassisches ABAP	wenig	**brauchbar, oft veraltete Syntax**
RAP, moderne CDS, EML	wenig	**schwach ohne Kontext**

Der ABAP-Code lebt im System

Bei anderen Sprachen liegt alles als Datei im Repository. Bei ABAP nicht.

	Andere Sprachen	ABAP
Wo liegt der Code?	als Datei im Repository	**im SAP-System**
Zugriff	jedes Werkzeug direkt	**nur über ADT (Eclipse, VSC)** (oder SAPGui)
Historie	Git, oft öffentlich	**Transporte, intern**
Was sieht die KI?	Alles im Projekt	**nur, was das Werkzeug holt**

Das Modell sieht euren Code nicht von selbst. **Deshalb geht es gleich um Kontext und MCP.**

Der Chat

Frage an Claude

Haiku 4.5 – 22.09.2026

Vorbereitet am 22.09.2026 – Antwort gekürzt

PROMPT

Wie heisst die Klasse/Methode in ABAP um einen Businesspartner zu löschen?

ANTWORT

Für das Löschen eines Businesspartners gibt es mehrere Optionen:

- **BAPI_BPARTNER_DELETE** – klassisch, aber deprecated
- OData Service **API_BUSINESS_PARTNER** mit **DELETE**-Operation
- **CL_BP_MANAGER** (aber ebenfalls veraltet)

Klingt souverän, stimmt aber überhaupt nichts! Wissensstand Feb. 2025.

Frage an Claude

Opus 5.5 – 22.09.2026

Vorbereitet am 22.09.2026 – Antwort gekürzt

PROMPT

Wie heisst die Klasse/Methode in ABAP um einen Businesspartner zu löschen?

ANTWORT

Es gibt keine freigegebene Klasse oder Methode, mit der man einen Business Partner direkt löscht. SAP sieht dafür Archivierung vor:

- Archivierungskennzeichen setzen mit **BAPI_BUPA_CENTRAL_CHANGE**
- Archivieren und physisch löschen über das Archivierungsobjekt **CA_BUPA in SARA**.
- Nur für Test- und Entwicklungssysteme gibt es die Transaktion **BUPA_DEL … BUP_PARTNER_DELETE**
- **In Clean Core und RAP stellen weder API_BUSINESS_PARTNER noch die BP-RAP-BOs eine Delete-Operation für den BP selbst bereit.**

Deutlich präziser und mit Einordnung, warum es kein Löschen gibt. Wissensstand Juni 2026.

Frage an Claude

Opus 5.5 – 22.09.2026

Vorbereitet am 22.09.2026 – Antwort gekürzt

PROMPT

Ich muss in ABAP die Adresse eines Business Partners ändern. Welche API soll ich dafür verwenden?

ANTWORT

Das hängt vom Entwicklungsmodell ab:

- Klassisches ABAP (On-Premise)
  - BAPI_BUPA_ADDRESS_CHANGE mit ADDRESSGUID aus BAPI_BUPA_ADDRESSES_GET, danach BAPI_TRANSACTION_COMMIT.
- ABAP Cloud / Clean Core
  - Released RAP-BO I_BusinessPartnerTP_3, Änderung per EML MODIFY ENTITIES. On-prem ab S/4HANA 2023. Remote: OData API_BUSINESS_PARTNER.
- Nicht empfohlen
  - CL_MD_BP_MAINTAIN=>MAINTAIN – verbreitet, aber nicht freigegeben.

Nicht falsch, aber unentschieden: Das Modell deckt beide Welten ab, weil es meine nicht kennt

Frage an Claude

Opus 5.5 – 22.09.2026

Vorbereitet am 22.09.2026 – Antwort gekürzt

PROMPT

Ich bin auf einem ECC 6.0 System, NetWeaver 7.40. Ändert das deine Empfehlung?

ANTWORT

Ja. Auf ECC 6.0 mit NW 7.40 gibt es **weder RAP noch I_BusinessPartnerTP_3**

- Empfehlung: **BAPI_BUPA_ADDRESS_CHANGE** – Adresse über ADDRESSGUID oder ADDRNUMBER, nur Felder mit X-Flag, danach BAPI_TRANSACTION_COMMIT.

- Vorher prüfen, welches Objekt führend ist:
  - BP mit CVI-Synchronisation → beim BUPA-BAPI bleiben
  - Klassischer Debitor/Kreditor ohne CVI → **CMD_EI_API** bzw. **VMD_EI_API**

Ein Satz Kontext, eine andere Antwort. Und ein Hinweis mehr, den die erste Antwort nicht hatte.

Demo 1 - Föß

Code im Chatfenster generieren

Womit arbeite ich?

Die Werkzeuge

Eclipse / ADT

Die offizielle ABAP-Entwicklungsumgebung

- Alle ABAP-Objekttypen
- Debugger, ATC, Transportwesen
- Der Ort, wo ihr ohnehin arbeitet
- Plugins und ADT MCP Server

VS Code / ADT

mit ABAP-Erweiterung

- Nur ein Teil der Objekttypen
- Starkes Umfeld: Git, Web, andere Sprachen
- Erste Wahl bei UI5
- Plugins und ADT MCP Server

CLI / Terminal

z. B. Claude Code

- Keine Bearbeitungsoberfläche
- Rein agentisch bedient
- Nah am Dateisystem

VS Code – der aktuelle Stand

Die Extension entwickelt sich schnell

Neue Objekttypen

Klassisches ABAP
Funktionsgruppen und -bausteine, Programme und Includes, Sperrobjekte

Data Dictionary
Datenbanktabellen, Strukturen, Typgruppen

RAP
Behavior Definition Extensions, CDS Entity Buffer

Neue MCP-Tools

Unified Diff für Transporte
„Erkläre Transport XYZ“ · „Wie riskant ist dieser Transport?“

ATC-Checks
„Implementiere X, dann ATC laufen lassen und die Findings fixen“

Dranbleiben:

Thomas Alexander Ritter (SAP) — Release-Blogs in der SAP Community. Folgen lohnt sich.

! ***Die Extension entwickelt sich schnell weiter!***

Joule for Developers: SAPs integrierter Weg

Stand September 2026. Seit Sapphire 2026 allgemein verfügbar, inkl. agentischer Funktionen

Was es kann

- Code Completion und Code-Erklärung
- Unit Tests für Klassen und CDS
- Objekte generieren, ATC-Befunde erklären
- agentisch, mit eigenem ABAP MCP Server

Voraussetzungen

- BTP ABAP Environment, S/4HANA Cloud Public und Private Edition ab 2021
- Lizenz über Joule Premium, Abrechnung in AI Units
- Aktivierung über SAP for Me, Ausführung über BTP

Einordnung

- Joule: SAP-optimiert, im Standard integriert
- Copilot, Claude Code: freie Modellwahl, sofort startklar
- Kein Entweder-oder: SAP unterstützt auch externe Agent-IDEs und LLMs

Wo KI im ABAP-Alltag bereits hilft

Nicht nur Code schreiben: Die größten Gewinne liegen oft beim Verstehen und Prüfen

01

Code verstehen

Legacy-Code, fremde Klassen und CDS-Hierarchien erklären lassen.

02

Refactoring

Clean ABAP, moderne Syntax, Clean-Core-Alternativen finden.

03

Unit Tests

Testklassen und Testdaten erzeugen, Randfälle vorschlagen.

04

Neue Objekte

RAP-BO-Gerüste, CDS Views, Behavior Implementations.

05

Qualität & Fehler

ATC-Befunde erklären und beheben, Dumps analysieren.

06

Doku & Research

Dokumentation und Specs schreiben

Kontext

Instructions: Regeln, die immer gelten

Einmal aufschreiben statt in jeden Prompt – das Tool schickt sie automatisch mit.

AGENTS.md (claude.md)

Wo & für wen

Kunde/Projekt, Namensraum, Systeme, Demo-Paket.
Eine Datei pro Workspace, tool-neutral.

*.instructions.md

Wie wir programmieren

Code-Standards ABAP. Mehrere Dateien, im User-Profil – gelten in jedem Kunden-Workspace.

So sieht das aus

```
# Namenskonventionen
Präfix ZCDX_ für alle Objekte

# ABAP-Regeln
Keine veraltete Syntax, keine
DDIC-basierten CDS Views
Nie direkt in SAP-Tabellen schreiben
Einfachste Lösung zuerst

# Sprache
Code und Kommentare auf Englisch
```

Mehr braucht es nicht für den Anfang. Vier, fünf Regeln ändern das Ergebnis deutlich.

Demo 2 - Föß

VS Code, Copilot und die erste AGENTS.md

Kontext für den KI-Kollegen

Das Modell kennt euer System nicht. Das Tool liefert ihm den Kontext: .md-Dateien sind sein Onboarding

Instructions

Wie arbeiten wir?

Regeln, Konventionen, Clean Core. Gelten dauerhaft.

AGENTS.md
wo & für wen · eine Datei

*.instructions.md
Programmierregeln · mehrere
z. B. general, abap, ui5

immer geladen

Skills

Was kann er noch?

Fachwissen für eine bestimmte Aufgabe, inkl. Vorlagen und Skripten.

SKILL.md
eigener Ordner je Skill

z. B. RAP-BO nach Cadaxo-Muster anlegen

bei Bedarf vom Tool geladen

Agenten

Was starte ich selbst?

Eigene Rollen und wiederkehrende Aufgaben.

*.agent.md
eigene Rolle & Tools
z. B. ATC-Reviewer

*.prompt.md
Aufruf per /name

per Aufruf

Specs

Was bauen wir?

Pro Feature gemeinsam mit der KI erarbeitet, dann umgesetzt.

spec.md
→ plan.md
→ tasks.md

Konvention, referenziert

Ebene 4: Zugriff - MCP-Server

Zugriff – Was die KI sieht und tun kann.

Wie die KI zugreift

Eingebaute Werkzeuge: Das Tool gibt dem Modell Hände.

Dateien

lesen, suchen, ändern – im Workspace

`read · grep · edit`

Shell & Git

Befehle im Terminal ausführen

`git diff · git commit`

Python

Skripte schreiben und laufen lassen

`python auswertung.py`

HTTP / Web

Seiten abrufen, im Web suchen

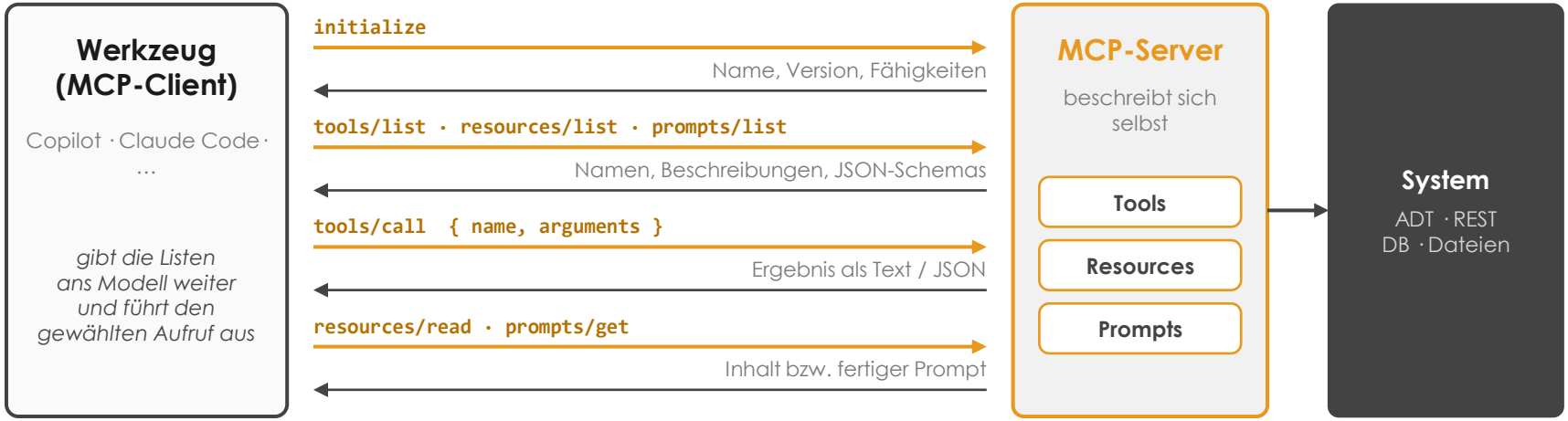
`fetch · web_search`

Eingebaut = fest im Tool. Je nach Tool unterschiedlich – der reine Chat kann z. B. kein Git im Workspace.
Der Agent arbeitet mit den Rechten des Users – kritische Aktionen lassen sich einzeln freigeben.

> ! ***Das Modell schreibt nur Text – ausführen tut immer das Tool***

MCP – Model Context Protocol

Das Werkzeug fragt den Server, was er kann – und ruft es dann auf.



Protokoll: JSON-RPC 2.0 · Transport: stdio (lokaler Prozess) oder Streamable HTTP (remote)

> ***Der Server beschreibt sich selbst – das Werkzeug muss ihn vorher nicht kennen***

Ein Blick auf tools/list

Echte Antwort von ARC-1 (gekürzt) – so beschreibt sich ein MCP-Server.

```
{
  "jsonrpc": "2.0", "id": 2,
  "result": { "tools": [
    {
      "name": "SAPRead",
      "description": "Read SAP ABAP objects – exact raw
         source, a method body, grep output, … Types:
         PROG, CLAS, INTF, FUNC, DDLS, BDEF, …",
      "inputSchema": {
        "type": "object",
        "properties": {
          "type":   { "type": "string",
                      "enum": ["PROG", "CLAS", "DDLS", …] },
          "name":   { "type": "string",
                      "description": "Object name …" },
          "method": { "type": "string", … }
        },
        "required": ["type"]
      },
      "annotations": { "readOnlyHint": true }
    },
    { "name": "SAPWrite", …, "annotations":
        { "readOnlyHint": false, "destructiveHint": true } },
    … SAPSearch, SAPActivate, SAPDiagnose, SAPTransport, SAPGit …
  ] }
}
```

`name`

Kennung – damit ruft das Werkzeug später tools/call auf

`description`

Freitext fürs Modell: danach entscheidet es, welches Tool passt

`inputSchema`

JSON-Schema der Parameter – das Modell füllt es aus

`annotations`

Hinweise ans Werkzeug: lesend oder zerstörerisch → Freigabe

Alle Beschreibungen landen bei jedem Aufruf im Kontext – gute Texte = gute Tool-Wahl

Zwei MCP-Server für ABAP

Beide nutzen die ADT-Schnittstelle, unterscheiden sich aber in Herkunft und Werkzeugen

SAP ADT MCP

offiziell von SAP

Kommt mit	ADT für VS Code und Eclipse (ab ADT 3.60)
Stärken	ATC-Checks, Transport-Diff, laufend neue Objekttypen
Läuft	Lokal in der IDE
Einsatz	Standard, erste Wahl

ARC-1

Open Source, Community

Kommt mit	eigenständig, als npm-Paket oder Docker-Image
Stärken	breites Toolset – im Prinzip alles, was man mit ADT machen kann
Läuft	Lokal oder zentral (BTP, beim Kunden)
Einsatz	wenn der verfügbare Toolumfang des offiziellen Servers nicht ausreicht

Mögliche Reihenfolge: 1 Dateien im Workspace › 2 SAP ADT MCP › 3 ARC-1

Demo 3 - Domi

Eclipse, Copilot und MCP

Autonomie

Skills und Agenten: Können und Rollen

Instructions sagen, wie wir arbeiten. Skills bringen Können für eine Aufgabe, Agenten eine Rolle

Skills das Handbuch für eine Aufgabe

Aufbau	Ordner mit SKILL.md, dazu Vorlagen, Beispiele, Skripte
Geladen	nur die Beschreibung ist immer bekannt, der Inhalt kommt bei Bedarf
Vorteil	Kontext bleibt schlank, gleiches Format in Copilot und Claude Code

Beispiel RAP-BO nach Kunden-Muster: Namen, CDS- und BDEF-Vorlagen, Checkliste

Agenten der Kollege mit eigener Rolle

Aufbau	*.agent.md mit Rolle, Anweisungen, Tool-Auswahl, ggf. Modell
Gestartet	vom Entwickler ausgewählt oder vom Haupt-Agenten beauftragt
Vorteil	klare Rollen, Werkzeuge gezielt einschränken

Beispiel ATC-Reviewer: darf lesen und ATC ausführen, aber nichts ändern

MCPs in der SAP-Welt: ein Abriss

Dutzende Server von SAP und Community, sortiert danach, was sie der KI geben

Wissen

bringt aktuelles SAP-Wissen ins Modell

SAP

- UI5 MCP
- CAP MCP
- Fiori MCP
- UI5 Web Comp.
- MDK MCP

Community

- MCP SAP Docs
- SAP Notes MCP
- BTP Docs MCP

Entwicklung

unser Fokus

liest und schreibt das ABAP-Repository über ADT

SAP

- SAP ADT MCP

Community

- ARC-1
- Vibing Steampunk
- mcp-abap-adt
- ABAP MCP SDK

Daten & Betrieb

Geschäftsdaten, Plattform und Integration

SAP

- BTP Administration
- LeanIX

Community

- OData-Bridges
- Integration Suite
- HANA
- Datasphere
- SAP GUI Scripting

Faustregel: Wissens-Server machen das Modell klüger, Entwicklungs-Server geben ihm Hände.

ARC-1 und die anderen

Drei Community-Server für ABAP, drei unterschiedliche Ideen

	ARC-1 `arc-mcp/arc-1`	Vibing Steampunk `oisee/vibing-steampunk`	ABAP MCP SDK `abap-ai/mcp`
Technik	TypeScript, npm-Paket oder Docker-Image	Go, ein einzelnes Binary	ABAP, per abapGit ins System
Läuft	lokal oder zentral: Docker, BTP Cloud Foundry	lokal auf dem Rechner	im SAP-System selbst, über ICF
Werkzeuge	12 Tools nach Absicht: SAPRead, SAPWrite, SAPDiagnose …	je nach Modus 1, 100 oder 147 Tools	eigene Tools, in ABAP geschrieben
Stärke	Enterprise: Anmeldung, Principal Propagation, Audit	Debugger, RFC-Aufrufe, Paketanalyse	kein externer Server, SAP-Berechtigungen nativ
Sicherheit	standardmäßig nur lesend, alles per Flag freischalten	--read-only, --allowed-packages	legt der Entwickler im Code fest

Unsere Wahl für Claude: ARC-1 für Teams und Betrieb · vsp für Power-User · ABAP SDK für eigene Tools

ARC-1: fünf Wege ins Deployment

Vom Laptop bis zur BTP. Die entscheidende Frage: Wer bin ich im SAP-System?

	Am Arbeitsplatz · ein Entwickler, stdio			Zentral · viele Nutzer, HTTP	
	Lokal npx	**Claude Desktop** MCPB	**Claude Code** Plugin	**Docker** Team-Server	**BTP** Cloud Foundry
Installation	`npx arc-1@latest`	`.mcpb per Doppelklick`	`/plugin install arc-1`	`Image von ghcr.io`	`MTA-Deployment`
Anmeldung an ARC-1	– lokaler Prozess	– lokaler Prozess	– lokaler Prozess	API-Key oder OIDC (Entra ID …)	XSUAA
Wer bin ich in SAP?	mein User	mein User	mein User	ein technischer User	**mein User, per Principal Propagation**
Gut für	schnell ausprobieren	Einstieg ohne Konsole	Entwickler, inkl. SAP-Skills	Team, schneller Start	Enterprise, Audit

Gleicher Server, gleiche Leitplanken: Es ändert sich nur, wo ARC-1 läuft und wer es im SAP ist.

Am Arbeitsplatz: in fünf Minuten verbunden

Lokal startet der Client den Server. Zentral kennt der Client nur eine URL

.mcp.json · lokal über stdio — startet ARC-1 selbst

```
{
  "mcpServers": {
    "arc-1": {
      "command": "npx",
      "args": ["-y", "arc-1@latest"],
      "env": {
        "SAP_URL":      "https://127.0.0.1:44300",
        "SAP_USER":     "DEVELOPER",
        "SAP_PASSWORD": "${SAP_PW}",
        "SAP_CLIENT":   "100"
      }
    }
  }
}
```

Terminal · zentral oder per Bundle — verbindet sich nur

```
# Team-Server im Netz (Docker / BTP)
claude mcp add --transport http \
  arc-1 https://arc1.intern/mcp \
  --header "Authorization: Bearer $KEY"

# Claude Desktop
# arc-1.mcpb aus den Releases laden,
# Doppelklick, Zugangsdaten eingeben

# Claude Code: Server + Skills
/plugin marketplace add arc-mcp/arc-1
/plugin install arc-1@arc-1
```

Tipp: Passwort per Umgebungsvariable, nie in die Datei. Dann darf die .mcp.json ins Git.

Zentral: Wer bin ich im SAP-System?

Beide Varianten bedienen viele Nutzer. Der Unterschied liegt in der Identität

Docker

auf einer VM im Firmennetz

Anmeldung per API-Key oder OIDC. ARC-1 kennt den Menschen, SAP sieht nur den technischen User. Einfach, aber ohne Audit-Trail im SAP.

BTP

Cloud Foundry mit Principal Propagation

Jeder arbeitet mit seinem eigenen SAP-User. Berechtigungen und Änderungsprotokoll wie in Eclipse. BTP ABAP und S/4 Public Cloud: Token-Exchange statt Cloud Connector.

! **Merke:** „Der Agent darf nur, was der User darf“ gilt sauber nur mit eigenem User.

Leitplanken, egal wo ARC-1 läuft

Was nicht freigeschaltet ist, kann der Agent nicht, egal was das Modell will

Standardmäßig aus bewusst freischalten

⊗ **Schreiben**	`SAP_ALLOW_WRITES`
⊗ **Freies SQL**	`SAP_ALLOW_FREE_SQL`
⊗ **Tabelleninhalte**	`SAP_ALLOW_DATA_PREVIEW`
⊗ **Transporte ändern**	`--allow-transport-writes`
⊗ **Git-Push, gCTS**	`--allow-git-writes`

Zusätzlich eingrenzen

Pakete	nur schreiben in "Z*,$TMP"
Aktionen	einzelne sperren, z. B. SAPWrite.delete
Profile	viewer, developer, admin je API-Key
Protokoll	Audit-Log in Datei oder BTP Audit Log Service

! **Die Server-Einstellung ist die Obergrenze:** Profile und Scopes können nur weiter einschränken.

Demo 4 - Sören

Agentische Entwicklung

Aus der Praxis

Was gut klappt, was nicht

Unsere Erfahrungen aus der Praxis. Je besser der Kontext, desto besser das Ergebnis

Klappt gut

- ✓ Code erklären, auch alten und fremden Code
- ✓ Unit Tests und Testdaten
- ✓ Refactoring auf moderne Syntax
- ✓ RAP- und CDS-Gerüste mit guten Instructions
- ✓ ATC-Befunde abarbeiten

Klappt (noch) nicht gut

- ✗ Veraltete Syntax, wenn keine Regel sie verbietet
- ✗ Fachlogik ohne Kontext und Spezifikation
- ✗ Inline-Code-Completion: passt meist nicht

Gegenmittel: **Regeln in der AGENTS.md, z. B. „keine direkten Tabellenänderungen“, „einfachste Lösung zuerst“.**

Läuft, aber nicht schön

Acht Korrekturen aus einem RAP-Projekt mit KI über ADT, drei Ursachen

A Framework

RAP-Lebenszyklus nicht verinnerlicht

1. Ableitung im Saver statt Determination
2. Nummern von Hand statt Early Numbering
3. Action in der Schleife statt im Batch

Gegenmittel
Skills mit RAP-Mustern, Review durch Erfahrene

B Projektkontext

weiß nicht, was es schon gibt

4. Basistabellen statt vorhandener Views
5. Freitext statt Message-Klasse

Gegenmittel
AGENTS.md: Views, Message-Klasse, Konventionen

C Stil und Altlast

Gewohnheit aus Trainingsdaten

6. Lese-Loops statt line_exists
7. FOR ALL ENTRIES statt JOIN
8. Magic Numbers

Gegenmittel
Regeln in der AGENTS.md, dazu ATC

! **Läuft, aber falsch gebaut:** Syntaxcheck und Unit-Tests finden das nicht. Das Review schon.

Fall 1: Fachlogik im Saver

Die KI leitet ein Feld beim Speichern ab. RAP hat dafür die Determination

Vorher **Save-Phase: save_modified**

```
METHOD save_modified.
  LOOP AT create-amount ASSIGNING FIELD-SYMBOL(<new>).
    " Berechnungsart ableiten, mitten im Saver
    SELECT SINGLE calc_type FROM zcust_condtype
      WHERE cond_type = @<new>-cond_type
      INTO @DATA(calc_type).
    " ... an den Writer übergeben
  ENDLOOP.
ENDMETHOD.
```

✕ Die Save-Phase darf nicht fehlschlagen. Der Wert fehlt im Puffer, also auch für Validierungen und Simulation.

Nachher **Determination**

A · Framework

```
" BDEF
determination determineCalcType on modify
  { create; field CondType; }

" Behavior Implementation
METHOD determineCalcType.
  READ ENTITIES OF zr_contract IN LOCAL MODE
    ENTITY Amount FIELDS ( CondType )
    WITH CORRESPONDING #( keys ) RESULT DATA(amounts).
  " ... Berechnungsart aus dem Customizing bestimmen
  MODIFY ENTITIES OF zr_contract IN LOCAL MODE
    ENTITY Amount UPDATE FIELDS ( CalcType )
    WITH updates.
ENDMETHOD.
```

✓ Der Wert steht sofort im Puffer. save_modified liest nur noch CalcType und speichert.

Kernaussage: save_modified speichert nur. Ableiten macht die Determination.

Fall 1, Teil 2: Irrtum mit Überzeugung

Beim Nachtesten zieht die KI aus einem leeren Feld den falschen Schluss

Die Hypothese der KI

„Die Determination läuft bei IN LOCAL MODE nicht."

Anlass: Beim Nachtesten war der Diagnosewert leer. Die KI schloss daraus auf einen Fehler im Framework, ohne es als Annahme zu kennzeichnen.

✕ **falsch, aber sehr überzeugend**

Was wirklich passiert

1. **`MODIFY ENTITIES … CREATE`**
   legt einen CREATE-Eintrag im Puffer an
2. **Determination läuft**
   auch bei IN LOCAL MODE
3. **schreibt CalcType**
   in denselben CREATE-Eintrag, kein neues UPDATE

`create-amount`	`update-amount`
CalcType gefüllt	leer, und das ist richtig

! **Lehre:** Hypothesen dort messen, wo der Wert wirklich landet. Ein leeres Feld ist kein Framework-Fehler.

Fall 2: Nummern von Hand

Die KI vergibt Schlüssel per SELECT MAX. RAP hat dafür Early Numbering

Vorher **Aufrufer vergibt den Schlüssel**

```
" nächste Nummer selbst ermitteln
SELECT MAX( contract_id ) FROM zcontract
  INTO @DATA(last_id).

MODIFY ENTITIES OF zr_contract
  ENTITY Contract
  CREATE FIELDS ( ContractId Description )
  WITH VALUE #( ( %cid        = 'C1'
                  ContractId  = last_id + 1
                  Description = 'Neu' ) ).
```

✕ Zwei Nutzer, eine Nummer: MAX + 1 ist nicht sperrsicher. Jeder Aufrufer muss die Logik kennen.

Nachher **Early Numbering** A · Framework

```
" BDEF
define behavior for ZR_Contract early numbering

METHOD earlynumbering_create.
  LOOP AT entities INTO DATA(entity).
    cl_numberrange_runtime=>number_get(
      EXPORTING nr_range_nr = '01'
                object      = 'ZCONTRACT'
      IMPORTING number      = DATA(number) ).
    APPEND VALUE #( %cid       = entity-%cid
                    ContractId = number )
      TO mapped-contract.
  ENDLOOP.
ENDMETHOD.
```

✓ Das BO vergibt die Nummer selbst, über ein Nummernkreisobjekt, für jeden Aufrufer gleich.

Kernaussage: Schlüssel vergibt das Business Object, nicht der Aufrufer.

Fall 3: Action in der Schleife

Die KI ruft eine Action Satz für Satz auf. EML kann alle Schlüssel auf einmal

Vorher **ein EML-Aufruf pro Satz**

```
LOOP AT contracts INTO DATA(contract).
  MODIFY ENTITIES OF zr_contract
    ENTITY Contract EXECUTE release
    FROM VALUE #( ( %tky = contract-%tky ) )
    FAILED   DATA(failed)
    REPORTED DATA(reported).
ENDLOOP.
" failed/reported: nur noch der letzte Satz
```

✕ n Aufrufe, n Handler-Durchläufe. failed und reported werden jedes Mal überschrieben.

Nachher **ein Aufruf für alle**

A · Framework

```
MODIFY ENTITIES OF zr_contract
  ENTITY Contract EXECUTE release
  FROM CORRESPONDING #( contracts )
  FAILED   DATA(failed)
  REPORTED DATA(reported).

" failed/reported: vollständig für alle Sätze
```

✓ Ein Aufruf, der Handler bekommt alle Schlüssel und kann gesammelt lesen und prüfen.

Kernaussage: EML ist mengenorientiert. Schleifen um EML sind fast immer falsch.

Fall 4: Basistabelle statt View

Die KI liest die Tabellen direkt. Im Projekt gibt es dafür längst einen View

Vorher **Join über die Tabellen**

```
DATA(today) = cl_abap_context_info=>get_system_date( ).

SELECT c~contract_id, c~supplier, p~valid_to
  FROM zcontract AS c
  INNER JOIN zcontract_pos AS p
    ON p~contract_id = c~contract_id
  WHERE p~valid_to >= @today
  INTO TABLE @DATA(active).
```

✕ Join-Logik doppelt, Access Control (DCL) des Views wird umgangen, Tabellenänderungen brechen den Code.

Nachher **vorhandener View**

B · Projektkontext

```
DATA(today) = cl_abap_context_info=>get_system_date( ).

SELECT ContractId, Supplier, ValidTo
  FROM zi_contractitem    " gibt es schon
  WHERE ValidTo >= @today
  INTO TABLE @DATA(active).
```

✓ Eine Stelle der Wahrheit: modellierte Beziehungen, Berechtigungen und Felder aus dem View.

Kernaussage: Die KI kennt nur, was sie sieht. Welche Views es gibt, gehört in die AGENTS.md.

Fall 5: Meldung als Freitext

Die KI schreibt Meldungstexte direkt in den Code. Im Projekt gilt die Message-Klasse

Vorher **Freitext im Code**

```
APPEND VALUE #(
  %tky = contract-%tky
  %msg = new_message_with_text(
    severity = if_abap_behv_message=>severity-error
    text     = 'Lieferant fehlt' ) )
  TO reported-contract.

" oder: new_message( ) mit fremder
" Message-Klasse und fester Nummer
```

✕ Nicht übersetzbar, nicht per Verwendungsnachweis auffindbar, jede Stelle formuliert anders.

Nachher **Projektkonvention** B · Projektkontext

```
APPEND VALUE #(
  %tky = contract-%tky
  %msg = NEW zcx_contract(
    textid   = zcx_contract=>supplier_missing
    severity = if_abap_behv_message=>severity-error )
  %element-Supplier = if_abap_behv=>mk-on )
  TO reported-contract.
```

✓ Übersetzbar, typisierte Textkonstante, Meldung hängt am richtigen Feld.

Kernaussage: Konventionen stehen nicht im Modell. Sie gehören in die AGENTS.md.

Fall 6: Lese-Loop statt line_exists

Die KI prüft Existenz wie vor 20 Jahren. Ein Ausdruck reicht

Vorher **Schleife mit Merker**

```
DATA(found) = abap_false.
LOOP AT contracts INTO DATA(contract)
     WHERE contract_id = id.
  found = abap_true.
  EXIT.
ENDLOOP.

IF found = abap_false.
  " ... Fehlermeldung
ENDIF.
```

✕ Sechs Zeilen und eine Hilfsvariable für eine Ja/Nein-Frage.

Nachher **line_exists**

C · Stil und Altlast

```
IF NOT line_exists( contracts[ contract_id = id ] ).
  " ... Fehlermeldung
ENDIF.
```

✓ Eine Zeile, die sagt, was sie meint. Mit sortiertem oder Hash-Schlüssel auch schnell.

Kernaussage: Die KI schreibt, was sie am häufigsten gesehen hat. Modernes ABAP muss man einfordern.

Fall 7: FOR ALL ENTRIES statt JOIN

Der Klassiker aus den Trainingsdaten. Heute geht der Join gegen die interne Tabelle

Vorher **FOR ALL ENTRIES**

```
IF keys IS NOT INITIAL.
  SELECT contract_id, amount
    FROM zi_contract
    FOR ALL ENTRIES IN @keys
    WHERE contract_id = @keys-contract_id
    INTO TABLE @DATA(contracts).
ENDIF.
" ohne IF: leere keys = alle Sätze!
```

✕ Fallstrick leere Tabelle, Duplikate werden still entfernt, Paketierung durch den DB-Treiber.

Nachher **JOIN gegen @itab**

C · Stil und Altlast

```
SELECT c~contract_id, c~amount
  FROM zi_contract AS c
  INNER JOIN @keys AS k
    ON c~contract_id = k~contract_id
  INTO TABLE @DATA(contracts).

" leere keys = korrekt kein Ergebnis
```

✓ Ein SELECT, der Filter läuft komplett in der Datenbank, leere Tabelle liefert korrekt nichts.

Kernaussage: FOR ALL ENTRIES ist nicht falsch, aber selten die beste Wahl.

Fall 8: Magic Numbers

Kein KI-Problem, aber die KI macht es genauso wie wir

Vorher **Literale im Code**

```
IF amount-calc_type = 'A'.
  total = amount-value * amount-quantity.
ELSEIF amount-calc_type = 'B'.
  total = amount-value.
ENDIF.

" was war noch mal 'A'?
```

✕ Die Bedeutung von 'A' und 'B' steht nirgends. Ändert sich ein Wert, sucht man jede Stelle.

Nachher **sprechende Konstanten**

C · Stil und Altlast

```
" INTERFACE zif_contract_c
"   CONSTANTS: BEGIN OF calc_type,
"     per_unit VALUE 'A', fixed VALUE 'B',
"   END OF calc_type.

CASE amount-calc_type.
  WHEN zif_contract_c=>calc_type-per_unit.
    total = amount-value * amount-quantity.
  WHEN zif_contract_c=>calc_type-fixed.
    total = amount-value.
ENDCASE.
```

✓ Lesbar, an einer Stelle definiert, per Verwendungsnachweis auffindbar.

Kernaussage: Was wir von Kollegen verlangen, gilt auch für die KI: Regeln in die AGENTS.md.

Die KI schreibt, der Entwickler unterschreibt!

Generierter Code sieht oft perfekt aus. Fachlich richtig ist er deshalb noch nicht.

Acht Beispiele, gerade gesehen

Keines davon findet der Syntaxcheck.
Keines davon findet ein Unit-Test, den dieselbe KI geschrieben hat.

Und eines davon war kein Fehler im Code, sondern eine falsche Erklärung – sehr überzeugend vorgetragen.

Es sieht auf den ersten Blick nicht falsch aus.
Das ist das Problem.

Review wie bei Kollegen

- ✓ **Fachlogik**
  stimmt es mit der Anforderung überein?
- ✓ **Framework**
  Nutzt es die vorgesehenen Mittel?
- ✓ **Tests**
  Die Anforderung muss getestet werden, nicht der Code.
- ✓ **Projektkontext**
  Gibt es den?
- ✓ **ATC & Performance**
  autmatisch prüfen lassen

Zwei Richtungen, zwei Themen

Was schicke ich zum Modell hinaus, was darf der Agent im SAP System

Datenschutz **Was schicke ich zum Modell?**

- Firmen-Lizenz statt Privat-Account: kein Training mit euren Daten
- Kundencode nur mit Freigabe
- Keine Echtdaten, nur Entwicklungssysteme

Sicherheit **Was darf der Agent im System?**

- Nur Entwicklungssysteme anbinden
- Tool-Aufrufe bestätigen, Tools einschränken
- Transporte gibt der Mensch frei
- MCP-Server mit dem Kunden abstimmen

Den Agenten wie einen “Externen” behandeln: nur Entwicklung, begrenzte Werkzeuge, der Mensch gibt frei.

Und jetzt?

Geht das bei mir?

Die KI-Tools selbst sind releaseunabhängig. Entscheidend ist, ob ADT mit eurem System spricht.

Das geht

ADT für VS Code
läuft gegen dieselben Systeme wie ADT für Eclipse, on-premise ab NetWeaver 7.3 EHP1 SP04

Kein Backend-Upgrade
wer heute ADT in Eclipse nutzt, braucht keine Systemänderung

MCP-Server
nutzen dieselbe ADT-Schnittstelle: wo ADT geht, geht auch MCP

Die Grenzen

VS Code
noch nicht alle Objekttypen, außerhalb des RAP-UI-Service-Flows bleibt Eclipse das Hauptwerkzeug

Klassische Dynpros, Formulare, …
nur über eingebettetes SAP GUI, für die KI kaum erreichbar

Joule for Developers
nur in bestimmten Umgebungen und mit eigener Lizenz

Faustregel: Wo ADT verfügbar ist, gibt es eine wichtige technische Grundlage für KI-gestützte ABAP-Entwicklung. Welche Funktionen tatsächlich möglich sind, hängt vom verwendeten Tool, MCP-Server und Backend ab

Womit fange ich an?

Klein starten, im eigenen System, mit einem Werkzeug

1

Ausprobieren

Chat, Test- oder Trial-System

2

Werkzeug holen

Copilot oder Claude Code, dazu ADT für VS Code

3

Kontext geben

erste AGENTS.md mit Namensraum und Regeln

4

Klein starten

Code erklären, dann Tests, dann Refactoring

Tipps zum Setup

- Der ADT MCP-Server ist aus Sicherheitsgründen abgeschaltet und muss in den Einstellungen aktiviert werden
- Eclipse bleibt parallel für alles, was VS Code noch nicht kann

Was kostet das?

- ?

Was bedeutet das für uns ABAP Entwickler?

Routinearbeit wird weniger. Gefragt sind Verständnis und Urteilskraft.

Die KI übernimmt mehr

- Tippen und Boilerplate
- Gerüste und Standardmuster
- Tests, Doku, erste Analyse

Wir werden wichtiger bei

- Fachverständnis: Was soll der Code wirklich tun?
- Architektur und Clean Core
- Kontext bauen: Instructions, Skills, Specs
- Review und Verantwortung

Der Schwerpunkt unserer Arbeit verschiebt sich.

- Weniger: Wie schreibe ich diesen Code?
- Mehr: Was soll er tun? Ist das die richtige Architektur? Ist das fachlich korrekt?

Und die Junioren?

Die KI ist besonders stark, wenn jemand beurteilen kann, ob ihre Lösung richtig ist.

Das Paradox

Erfahrene werden mit KI schneller, weil sie sofort sehen, ob der Code taugt.

↓

Diese Urteilskraft entsteht nur durch eigenes Entwickeln: Fehler suchen, debuggen, verstehen.

↓

Wer das der KI überlässt, wird schneller fertig, aber nie Senior.

Für Junioren

- Tutor statt Ghostwriter: erklären lassen, nicht lösen lassen
- Nur Code übernehmen, den man selbst erklären kann
- Grundlagen bewusst ohne KI erarbeiten
- KI-Code gemeinsam mit Senioren reviewen

Für Firmen

Weiter Junioren ausbilden und Zeit fürs Lernen einplanen. Sonst fehlen in fünf bis zehn Jahren die Senioren, die KI-Code beurteilen können.

Deshalb müssen wir überlegen, wie wir Junioren mit KI ausbilden.

S/4FIT

Zum Mitnehmen

1 **Kontext entscheidet**
Instructions, Skills & Specs machen aus einem Modell einen brauchbaren Kollegen.

2 **MCP bringt die KI ins System**
SAP ADT MCP als offizieller Weg – weitere Möglichkeiten über andere MCP-Server.

3 **Wir bleiben verantwortlich**
KI-Code prüfen wie Code von einem Kollegen. Die KI schreibt. Der Entwickler unterschreibt.

4 **Heute anfangen**
Eigenes System. Kleine Aufgabe. Ein Werkzeug. Nicht auf das perfekte Setup warten.

Ihr wollt tiefer einsteigen?

ABAP & KI – Hands-on Workshops

Vom ersten KI-Experiment bis zur agentischen ABAP-Entwicklung.

- **Wien** · Cadaxo – **ABAP-KI-Intensivworkshop**
  5. November 2026 und 10. Dezember 2026
- **Mannheim** · Brandeis – **Crashkurs und Deep Dive**
  9. / 10. Dezember 2026, 18. / 19. Januar 2027
- **Inhouse** – beide, auch bei euch im Haus

Wien

Mannheim