Agent-Plugin für die MoWare-Werkbank

# MoAI

## Coding-Agenten, die **MoWare verstehen** und direkt im MPS-Modell arbeiten.

MoAI bündelt Agent-Skills, Sprachdokumentation und verbindliche Konventionen für die drei MoWare-DSLs ObjectFlow, ManMap und DataUX sowie erprobte Workflows für JetBrains MPS. Das Paket liegt als versioniertes Git-Submodul im Anwendungsprojekt und wird gleichzeitig als Plugin im Coding-Agenten registriert.

| | |
|---|---|
| Skills | 12 |
| JSON-Blueprints | 45 |
| Sprachdokumente | 4 |
| Konventionsdateien | 6 |
| Basis | MPS 2026.1 |

## So greift alles ineinander

ANWENDUNGSPROJEKT
**AGENTS.md + moai/**
Projektregeln aus der Vorlage, MoAI als Submodul in fester Version.

→

CODING-AGENT
**lädt moai:-Skills**
liest Dokumentation und Konventionen nur, wenn die Aufgabe sie braucht.

→

MPS MCP
**modellbewusste Tools**
Knoten lesen, per JSON-Blueprint einfügen, prüfen und bauen.

→

JETBRAINS MPS
**Live-Modelle**
ObjectFlow, ManMap und DataUX, generiert zur lauffähigen Java-Anwendung.

## Inhalt des Pakets

**skills/** 12 SKILLS
Regeln, Rezepte, Beispiele, Blueprints

**docu/** 4 DOKUMENTE
Werkbank-Architektur und die drei DSLs

**conventions/** 6 DATEIEN
Verbindliche Modellierungskonventionen

**MPS_AGENT_GUIDE.md**
Arbeitsregeln für MPS-Arbeit

**INSTALL_INSTRUCTIONS.md**
Installation, als Agent-Prompt formuliert

**TEMPLATE_PROJECT_AGENTS.md**
Vorlage für die AGENTS.md im Projekt

## Die Skills

MOWARE-SPRACHEN

| | |
|---|---|
| moai:objectflow-dsl | Domäne, Services, Commands |
| moai:manmap-dsl | Persistenz, Repositories, SQL |
| moai:datax-dsl | Oberflächen, Apps, Batchjobs |
| moai:baselanguage-collections-dsl | Collections |

MPS-HUB

| | |
|---|---|
| moai:mps-mcp-workflow | Einstieg und Skill-Index |

MPS-WERKZEUGE

| | |
|---|---|
| moai:mps-node-editing | Knoten per Blueprint |
| moai:mps-baselanguage | Java als AST |
| moai:mps-model-manipulation | smodel, closures |
| moai:mps-console | Konsole, Queries |
| moai:mps-language-analysis | Sprachen analysieren |
| moai:mps-run-configurations | Mains und Tests starten |

AUFBAU EINES DSL-SKILLS

**SKILL.md**
Kritische Regeln, Quick Start, Doku-Landkarte

**concepts**
Konzepte und geprüfte AST-Rollen

**workflows**
Rezepte zum Anlegen und Ändern

**sandbox**
Referenzbeispiele aus mitgelieferten Solutions

**blueprints**
JSON-Skelette mit TARGET_*-Platzhaltern

**gotchas**
Stolperfallen und Diagnose

## Installation in drei Schritten

01

### Submodul einbinden

Im Wurzelverzeichnis des Anwendungsprojekts:

```
git submodule add \
  https://github.com/
  danielstieger/moai.git moai
git submodule update \
  --init --recursive
```

MoAI steht in keinem Marketplace und wird nur lokal eingebunden.

02

### Agent installieren lassen

Die Anleitung ist als Prompt geschrieben, daher genügt:

> *„Lies INSTALL_INSTRUCTIONS.md aus ./moai und führe die Installation für dieses Projekt durch.“*

Der Agent registriert das Plugin projektbezogen und legt die `AGENTS.md` aus der Vorlage an.

03

### Prüfen

Der Agent prüft, ob `moai:mps-mcp-workflow` und alle weiteren Skills sichtbar sind.
Sein Abschlussbericht nennt Version, angelegte Dateien, MCP-Status und offene Schritte wie einen Neustart oder Commit.

**Voraussetzungen:** JetBrains MPS läuft mit geöffnetem Zielprojekt, die MPS-MCP-Integration ist aktiviert und das MoAI-Plugin ist in der Agent-Runtime installiert und eingeschaltet.

## So arbeitet der Agent

| | |
|---|---|
| 01 | **AGENTS.md lesen:** Paketgrenzen, Skills, Abweichungen |
| 02 | **Konventionen** aus `moai/conventions/` vollständig lesen |
| 03 | **MPS_AGENT_GUIDE.md**, dann `mps-mcp-workflow` laden |
| 04 | **DSL-Skill** der betroffenen Schicht laden |
| 05 | **Blueprint** wählen, Platzhalter ersetzen, als Dry-Run prüfen |
| 06 | **Einfügen** als Skelett, dann gezielte Teiländerungen |
| 07 | **Validieren** jede geänderte Root mit `check_root_node_problems` |
| 08 | **Bauen** Rebuild in MPS, Tests ausführen |

> **Gestuft lesen:** Alle vier Dokumente kosten etwa 140.000 Tokens. Für Recherche und Planung reichen die Werkbank-Übersicht und die Kapitellandkarten der DSLs.

## Spielregeln

- **Nie rohes MPS-XML** lesen oder ändern, nur MCP-Tools
- **Generierter Code** ist tabu, korrigiert wird im Modell
- **`moai/` bleibt unverändert**, Projektspezifisches liegt außerhalb
- **MoWare-Sprachen** werden genutzt, nie geändert
- **Datenbankschema** erstellt der Entwickler in MPS
- **MoAI ist der einzige Skill-Katalog**, keine Fremd-Skills
- **Fehler beheben**, Warnungen nicht wegmodellieren
- **Projektpfade ohne Leerzeichen** (MCP-Bug)

## Verbindliche Konventionen

Die Konventionen sind der MoWare-Standard. Ein Projekt kann einzelne davon in seiner `AGENTS.md` abändern; ein Agent weicht nie eigenmächtig ab.

### Allgemein

Deutsche Ubiquitous Language, Schema durch den Entwickler

### Anwendungsaufbau

Bounded Contexts, Bereiche, Solutions, Modelle, Importe

### Domänenmodell

Invarianten am Aggregat, Zustandsübergänge als Service Methods

### Commands

Fachliche Namen, Owner und Edit, Rückgabe an die Suche

### Oberfläche

Page Titles, volle Breite, Overflow-Menüs, Hotkeys

### Tests

Zustandsübergänge, plausible Testdaten, Custom Sessions

AKTUALISIEREN

Das Anwendungsprojekt legt die MoAI-Version über den Commit des Submoduls fest. Ein Update wird als neuer Stand des Submoduls versioniert:

```
git submodule update --remote --merge moai
git add moai
```

REPOSITORY & KONTAKT

### MoAI auf GitHub

REPO **github.com/danielstieger/moai**
WEB **www.modellwerkstatt.org**
MAIL **info@modellwerkstatt.org**

Stand Oktober 2026 · JetBrains MPS 2026.1