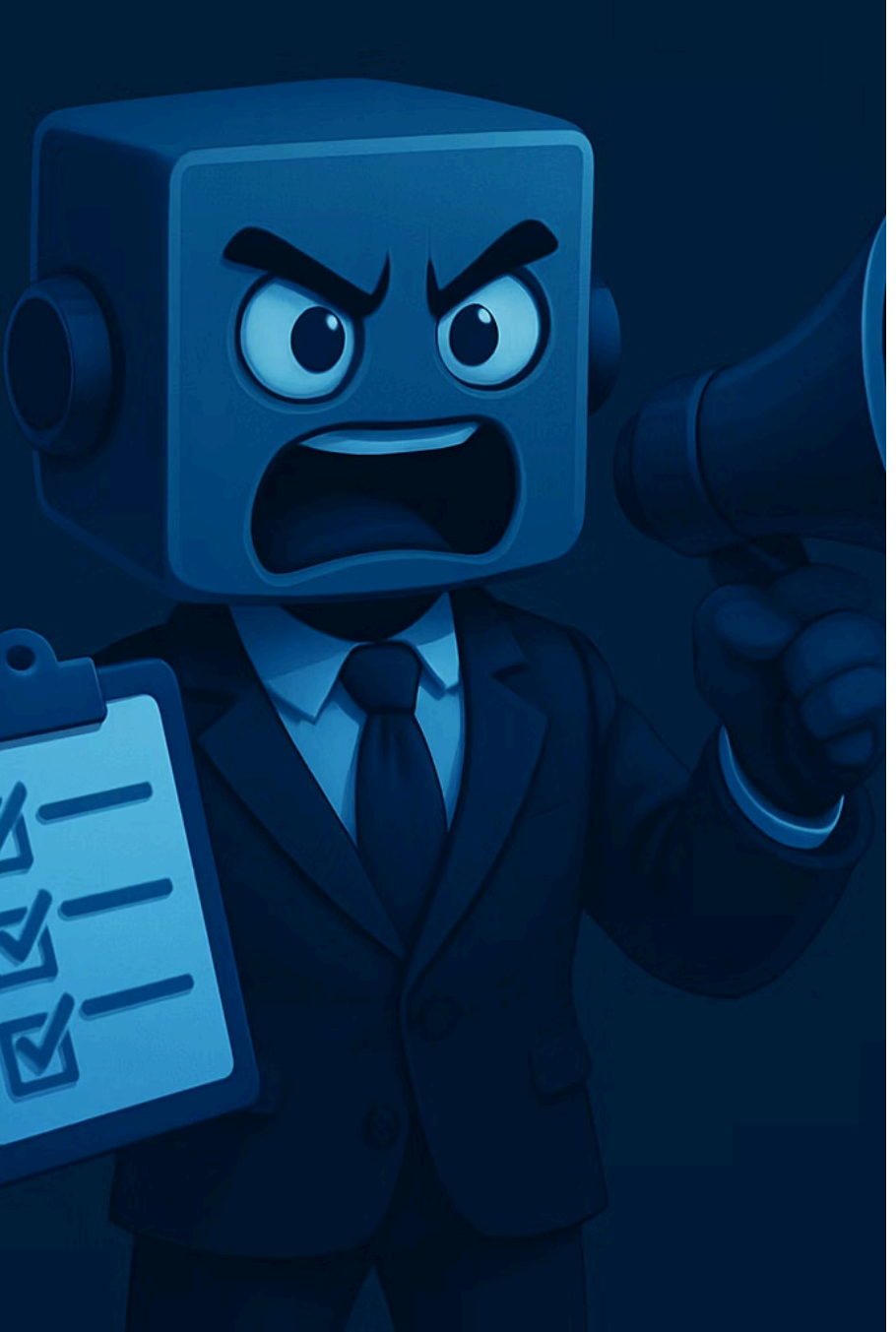


WOARU: Der Proaktive Meta-Qualitäts-Agent

Ein sprachagnostischer "Tech Lead in a Box", der direkt in der Entwicklungsumgebung läuft, die Codequalität in Echtzeit überwacht und proaktiv eingreift, um sicherzustellen, dass Projekte nicht nur syntaktisch korrekt, sondern auch nach Best Practices strukturiert sind.

Die "Last Mile" der KI-gestützten Entwicklung

Kontextverlust

KI-Assistenten vergessen schnell Linter-Regeln, Projekt-Konventionen oder den Gesamtkontext, was zu inkonsistentem oder fehlerhaftem Code führt.

Verzögerter Feedback-Loop

Qualitätsmängel werden oft erst spät in der CI-Pipeline oder bei Code-Reviews entdeckt. Dieser späte Kontextwechsel ist teuer und ineffizient.

Architektonische Blindheit

Entwickler und KIs konzentrieren sich auf die Implementierung von Features, vernachlässigen aber oft übergeordnete Best Practices wie Error-Monitoring, Testing-Infrastruktur oder sicheres Konfigurationsmanagement.

Bestehende Tools wie Linter-Plugins oder Pre-Commit-Hooks lösen diese Probleme nur teilweise. Plugins sind oft "zu leise", und Hooks sind "zu spät".

Vision & Lösung: Der Proaktive Meta-Qualitäts-Agent

WOARU agiert als permanenter, wachsamer Prozess im Hintergrund der lokalen Entwicklungsumgebung. Seine Aufgabe ist es, die Schwächen des modernen Entwicklungs-Workflows auszugleichen.

WOARU verfolgt einen dreistufigen Ansatz, der Audit & Bootstrap, Live-Intervention und universelle Anwendbarkeit kombiniert, um eine durchgängige Qualitätssicherung zu gewährleisten.

Audit & Bootstrap: Der Architekt

Projekt-Analyse

Beim Start auditiert WOARU das Projekt: Welche Sprachen und Frameworks werden verwendet? Welche Qualitäts-Tools sind bereits konfiguriert?

Auto-Konfiguration

Fehlen essenzielle Konfigurationen, kann WOARU sie basierend auf etablierten Best Practices erstellen (z.B. eine strenge .eslintrc.js für ein TypeScript-Projekt).

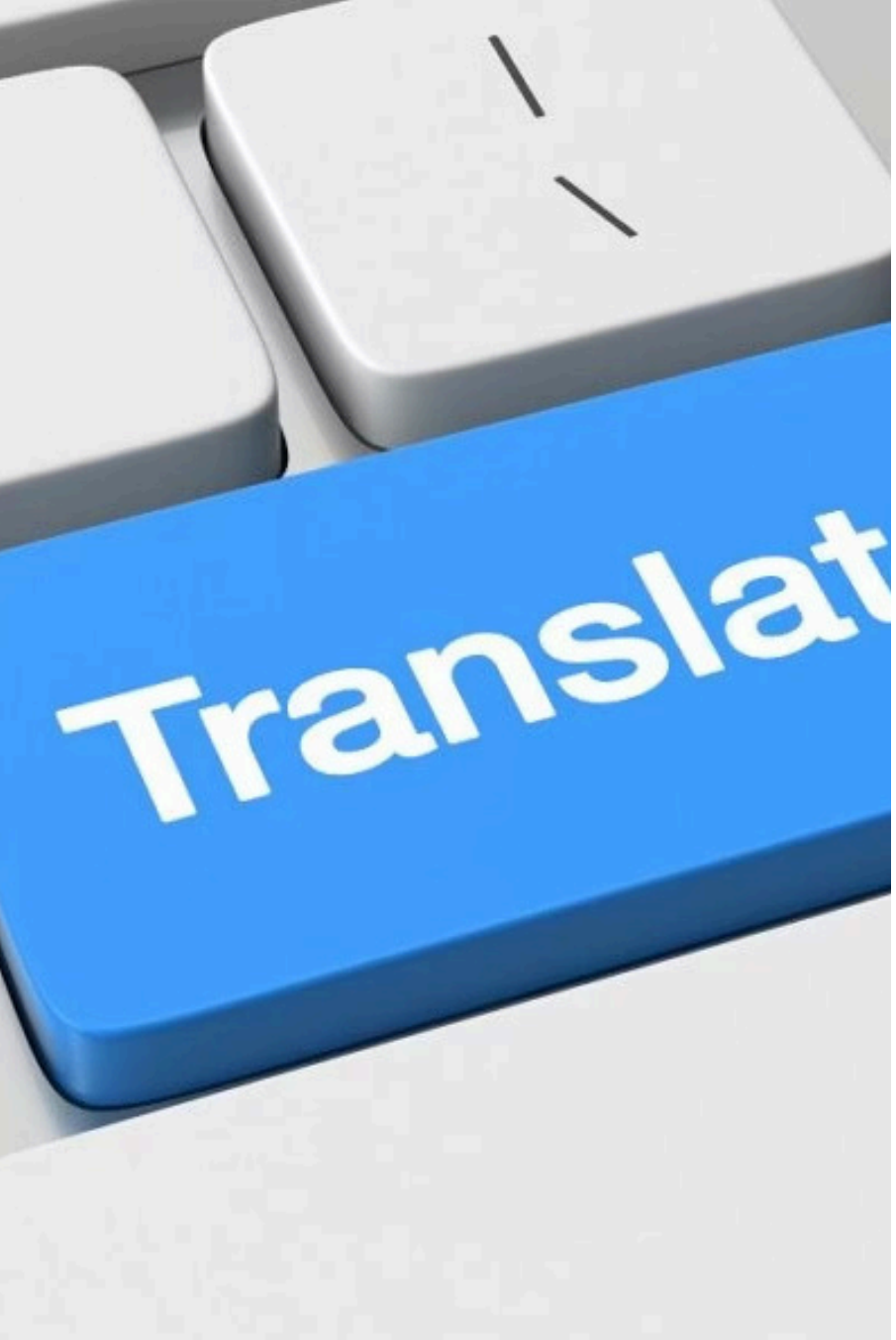
Produktionsreife-Prüfung

WOARU prüft, ob wichtige Produktions-Tools wie Sentry, Jest/Pytest oder eine Dockerfile vorhanden sind und gibt proaktive Empfehlungen.

Live-Intervention: Der Aufpasser

- Echtzeit-Monitoring: WOARU überwacht das Dateisystem. Sobald eine Datei gespeichert wird, wird sofort gehandelt.
- Sofortige Qualitäts-Checks: Basierend auf dem Dateityp wird das passende Tool ausgeführt (ESLint für .ts, Ruff für .py, gofmt für .go etc.).
- Lautes, unübersehbares Feedback: Schlägt ein Check fehl, gibt WOARU eine unmissverständliche, detaillierte Fehlermeldung im Terminal aus.

Dieser "harte" Feedback-Loop zwingt zur sofortigen Korrektur und ist für KI-Assistenten direkt lesbar und verarbeitbar.



Universelle Anwendbarkeit: Der Universalübersetzer

Sprachagnostische Architektur

Der Kern von WOARU ist modular aufgebaut (QualityRunner). Die Unterstützung für neue Sprachen und deren spezifische Linter/Formatter kann einfach durch neue Konfigurationen hinzugefügt werden.

Ein Tool für alle Projekte

Das Ziel ist ein einziges, via npx woaru watch aufrufbares Tool, das in jedem beliebigen Projekt sofort einen Mehrwert liefert, unabhängig vom Technologie-Stack.

Technische Architektur

WOARU ist eine in TypeScript geschriebene Node.js-Anwendung mit folgenden Kernkomponenten:

WOARUSupervisor

Die zentrale Orchestrierungs-Instanz. Startet und verwaltet alle anderen Module.

ProjectAnalyzer

Erkennt Sprachen, Frameworks und Abhängigkeiten durch Analyse von package.json, pyproject.toml, Dateiendungen etc.

NotificationManager

Verantwortlich für die formatierte und "laute" Ausgabe von Erfolgs- und Fehlermeldungen im Terminal sowie optional auf dem Desktop.

FileWatcher

Ein performanter, Event-basierter Dateisystem-Watcher, der node_modules, dist etc. intelligent ignoriert.

QualityRunner

Das Herzstück der Live-Intervention. Enthält die Logik, um bei einer Dateiänderung das korrekte externe Tool aufzurufen.

ProductionReadinessAuditor

Ein zukünftiges Modul, das strategische Audits durchführt (Sentry-Check, etc.) und Empfehlungen ausspricht.

Strategischer Nutzen

Steigerung der Code-Qualität

Erzwingt einheitliche Standards im gesamten Team, bevor der Code überhaupt committed wird.

Förderung von Best Practices

Macht Entwickler proaktiv auf architektonische Verbesserungen aufmerksam und fungiert als Lehrmittel.

Effizienzsteigerung

Reduziert dramatisch den Zeitaufwand für das Beheben von Linter-Fehlern in der CI/CD-Pipeline oder während Code-Reviews.

Senkung der Einstiegshürde

Neue Teammitglieder können sofort produktiv arbeiten, da WOARU die Einhaltung der Regeln sicherstellt.

Verbesserte KI-Kollaboration

Dient als externes "Gedächtnis" für KI-Assistenten, das sie auf Kurs hält und ihre Schwächen kompensiert.

Aktueller Status

Kernfunktionalität für JavaScript/TypeScript implementiert. Phase 1 abgeschlossen, Phase 2 (Erweiterung auf Python) beginnt.

Produkt-Roadmap: WOARU

Die Entwicklung von WOARU ist in vier strategische Phasen unterteilt, die aufeinander aufbauen und jeweils signifikanten Mehrwert liefern.

1

✅ Phase 1: Live-Qualitäts-Agent

Ziel: Robuster, sofortiger Feedback-Loop. **Status:** Abgeschlossen.

- Stabile TypeScript/Node.js-Architektur mit performantem File-Watcher.
- Modularer QualityRunner und "laute" Fehlerbenachrichtigungen.
- Vollständige Implementierung für JavaScript/TypeScript (ESLint).

2

🚀 Phase 2: Universal-Agent

Ziel: Sprachagnostischer Assistent. **Status:** In Umsetzung.

- Python-Support (ruff) in Arbeit.
- Geplant: Go, Rust und Web-Frontend (CSS/HTML) Support.
- Zukünftig: Konfigurierbarer Runner für eigene Regeln.

3

🧠 Phase 3: Produktionsreife-Auditor

Ziel: Architektonische & strategische Projektberatung. **Status:** Konzeption abgeschlossen.

- Entwicklung des ProductionReadinessAuditor-Moduls.
- Audits für Error-Monitoring, Testing-Frameworks, Infrastruktur.
- Security- & Konfigurations-Audit.

4

🌐 Phase 4: Das Ökosystem

Ziel: WOARU als zentrales Werkzeug etablieren. **Status:** Langfristige Vision.

- Veröffentlichung auf npm mit Auto-Bootstrap-Funktion.
- Editor-Integration (VS Code-Plugin) und optionale Dashboard-UI.