Warum der Wandel?

Status Quo

Feature → Ticket → Code → PR → Review

70-80% mechanische Tätigkeiten

Problem

Entwickler verbringen Zeit mit repetitiven Aufgaben statt mit Systemdesign und Architekturentscheidungen

Neue Engpässe

- Entscheidungsqualität
- Systemdenken
- Testbarkeit

Fokuswechsel

Von "Code schreiben" zu "Systeme entwerfen, die Code erzeugen"

Executive Summary

1

Der Paradigmenwechsel

Von Code zu Prozess: Die Zukunft liegt im Systemdesign

2

Kernthese

Agilität für Menschen = Kommunikation, für KI = Explizitheit

3

Die neue Regel

Was nicht explizit ist, existiert für KI nicht

4

Review-Shift

Entscheidungen prüfen, dann generieren

Duale Agilität

Mensch ↔ Mensch

Gespräche

Workshops

Aushandlung

Mensch ↔ Maschine

Spezifikationen

Regeln (bindend, versioniert)

Artefakte

KI ist nicht interpretierend, nicht aushandelnd

STAKEHOLDER-SHIFT

Verantwortung rückt nach vorne

Implizit → Explizit

Zentrale Transformation

Artefakte als Wahrheit zwischen Mensch und Maschine

Git

Normativ

Confluence

Warum

Jira

Arbeit steuern

Explizite Regeln • Invarianten • Verträge

SDD Kernprinzipien

1. Specs sind Single Source of Truth
2. Code implementiert Specs, ist niemals selbst die Spec
3. Verhaltenänderungen erfordern Spec-Änderungen
4. Qualität lokal deterministisch prüfen
5. Tests als ausführbare Spezifikationen

Arten von Specs

Architektur-Specs

`Constitution`

Systemweite, nicht verhandelbare Regeln

System/Feature-Specs

`Behavior`

Verhalten, Zustände, Schnittstellen

Komponenten/Modul-Specs

`Identity`

Identität, Inputs/Outputs, Fehlerfälle

Use Cases & Akzeptanzkriterien

`Testable`

Testbare Verhaltensanforderungen

Spec-Driven Development

KI-fähige, spec-getriebene Softwareentwicklung

Von der Implementierung zur Prozessgestaltung

Multi-Agent-Workflow

Architect

Spezifikation, ADRs, Annahmen

Planner

Tasks mit Akzeptanzkriterien

Implementer

Minimale Code-Umsetzung

Tester

Tests schreiben, Gates grün machen

Reviewer

Review gegen Specs, findings dokumentieren

Artefakt-Handoffs statt Chat-Handoffs

SDD vs. Wasserfall

Wasserfall

- Große, monolithische Specs
- Späte Validierung
- Änderungen teuer
- Statische Specs

SDD

- Kleine Spec-Änderungen
- Sofortige Ableitungen
- Iterativer Regelkreis
- KI macht Varianten praktikabel

Spec → Code → Tests → Feedback → Anpassung

Qualitätsschranken (Gates)

Eigenschaften

- Format, Lint, Tests lokal ausführbar
- Agent-neutral und deterministisch
- Task ist "fertig" nur wenn alle Gates grün

Automation Mode

Patches zwischen Agenten

State Machine

→ Spec

→ Plan

→ Tasks

→ Impl/Test-Loop

→ Review

→ PR

Review-Shift

Neue Reihenfolge

Spec Review → Generierung → Abweichungsprüfung → Code Review

LEITFRAGE

Entspricht der Code der Spec?

"Code-Reviews ohne Specs sind Meinungen, keine Qualitätssicherung"

Repo-Struktur

- specs/
  - constitution.md
  - architecture/
  - features/
- .github/
  - agents/
  - copilot-instructions
- scripts/
  - gates/
  - prompts/
  - workflow orchestrator

Git als normative Quelle

POLICY

No Spec, no merge

Brownfield-Strategie

SpecKit

Leichtgewichtig, inkrementell, feature-lokal

Bestandteile

- Mini-Spec
- Constraints
- Tasks

> "Fix dieses Verhalten, brech X nicht"

Kein Rollenspiel-Overhead

BMAD zu schwer für Brownfield

30-Tage-Einführung

1 Woche 1

Constitution + 1-2 kritische Use Cases

2 Woche 2

Tests aus Specs ableiten, Gates verdrahten

3 Woche 3

Preview pro Branch, Review-Shift einführen

4 Woche 4

Kleines Feature mit Copilot-Workflow

METRIKEN

Zeit bis Preview • Spec-Änderungen/Woche • Rework-Quote • Gate-Stabilität

Anti-Patterns & Zusammenfassung

Zu vermeiden

- Prompten ohne Spec
- Confluence als Wahrheit
- Eingefrorene Specs

Nächste Schritte

Mit kleinem Feature starten

Iterieren

Lernen

Zusammenfassung

- Explizitheit ist der Schlüssel
- KI als Umsetzer nicht als Teilnehmer
- Specs leben und lernen, nicht einfrieren


KI-fähige, spec-getriebene Softwareentwicklung


Executive Summary


Warum der Wandel?


Duale Agilität


Implizit → Explizit


SDD Kernprinzipien


Acten von Specs


SDD vs. Wasserfall


Multi-Agent-Workflow


Qualitätssicherungsketten (Gates)


Review-Shift


Repo-Struktur


Armenfield-Strategie


30-Tage-Einführung


Anti-Patterns & Zusammenfassung