
WHITEPAPER · AI ENABLEMENT & AGENTIC DELIVERY

Spec-Driven und Behavior-Driven Development

HERAUSGEBER

Manuel Fernandez

VERÖFFENTLICHT

15. Oktober 2026

01 Executive Summary

Viele Vorhaben scheitern nicht an der Technik, sondern an unklaren Anforderungen. Behavior-Driven Development (BDD) und Spec-Driven Development (SDD) machen fachliche Erwartungen zu überprüfbaren Spezifikationen – als gemeinsame Grundlage für Design, Umsetzung, Test und Abnahme.

BDD beschreibt das gewünschte Verhalten eines Systems gemeinsam mit Fachbereich, Entwicklung und Test anhand konkreter Beispielszenarien im Format Given/When/Then. **SDD** macht Spezifikationen – Szenarien, Geschäftsregeln, Schnittstellen, nicht-funktionale Anforderungen – zur verbindlichen, versionierten Grundlage der Delivery: Verhaltensänderungen beginnen mit einer Änderung der Spezifikation, nicht im Code.

Mit KI-Coding-Agenten gewinnt dieser Ansatz an Gewicht. Wenn Code schneller entsteht, wird die Frage, was genau entstehen soll und woran man erkennt, dass es richtig ist, zum Engpass. Unsere Kernthese: Spezifikationen sind das eigentliche Steuerungsinstrument moderner, auch KI-unterstützter Delivery. Wer sie gemeinsam erarbeitet, automatisiert prüft und an dokumentierte menschliche Freigaben koppelt, behält Transparenz und Kontrolle – unabhängig davon, wer oder was den Code schreibt.

Das Whitepaper richtet sich an Product Owner, Delivery Leads, Qualitätssicherung und Architektur. Es beschreibt die typischen Probleme, den Aufbau ausführbarer Spezifikationen, die Rolle von KI-Coding-Agenten, Traceability und Quality Gates, ein Vorgehen in sechs Phasen, die wesentlichen Artefakte sowie eine Checkliste für den Start.

02 Problemstellung

Eine Anforderung durchläuft meist mehrere Übersetzungsschritte: Der Fachbereich formuliert ein Anliegen, die Business-Analyse ein Konzept, die Entwicklung interpretiert es, die Qualitätssicherung leitet eigene Testfälle ab, und die Abnahme prüft gegen eine Erwartung, die bis dahin oft niemand mehr genau kennt. Typische Folgen:

- **Unschärfe Anforderungen:** Grenzfälle, Ausnahmen und Fehlerpfade werden erst im Test entdeckt.
- **Getrennte Wahrheiten:** Dokument, Ticket, Testfall und Code beschreiben dasselbe Verhalten jeweils anders.
- **Späte Abnahmekonflikte:** Unterschiedliche Annahmen zeigen sich in der teuersten Projektphase.
- **Fehlende Nachvollziehbarkeit:** Welche Anforderungen ein Release nachweislich erfüllt, lässt sich nicht schnell belegen – in regulierten Umfeldern ein Compliance-Thema.

- **Tests ohne fachlichen Bezug:** Automatisierte Tests prüfen Technik, aber nicht erkennbar die Geschäftsregeln.

KI-gestützte Entwicklung verschärft diese Punkte. Generierte Ergebnisse wirken oft vollständig und konsistent – gerade deshalb fallen fachliche Abweichungen ohne präzise Prüfkriterien schwerer auf. Und die Frage, wer was auf welcher Grundlage freigegeben hat, wird ohne explizite Spezifikationen noch schwerer zu beantworten.

Das Grundproblem ist dabei nicht mangelnde Sorgfalt einzelner Beteiligter, sondern das Fehlen eines gemeinsamen, überprüfbaren Bezugspunkts. Solange jede Rolle ihre eigene Fassung der Anforderung pflegt, bleibt die Abstimmung Handarbeit – und Klärung findet dort statt, wo sie am meisten Nacharbeit verursacht.

LEITFRAGEN

- Nennen Fachbereich und Entwicklung für eine zentrale Geschäftsregel dieselben Beispiele?
- Lässt sich je Release zeigen, welche Anforderungen durch welche Tests abgedeckt sind?
- Wer entscheidet heute, ob ein KI-generiertes Ergebnis fachlich korrekt ist – und anhand welcher Kriterien?

03 Ausführbare Spezifikationen mit Gherkin

BDD wurde ab Mitte der 2000er-Jahre von Dan North als Weiterentwicklung testgetriebener Entwicklung beschrieben. Es ist zuerst eine Klärungsmethode: In kurzen Gesprächen zu dritt („Three Amigos“: Fachlichkeit, Entwicklung, Test) werden zu einer Story Regeln, Beispiele und offene Fragen gesammelt, etwa per Example Mapping. Die Beispiele werden in Gherkin notiert, der Beschreibungssprache des Cucumber-Ökosystems: **Given** (Ausgangssituation), **When** (Aktion), **Then** (beobachtbares Ergebnis); auf Deutsch „Angenommen“, „Wenn“, „Dann“. Über Step Definitions wird jeder Schritt mit Testcode verbunden – die Spezifikation ist dann entweder erfüllt oder nicht.

```

# language: de
# Hypothetisches Beispiel – keine reale Kundenanforderung
Funktionalität: Freigabe von Bestellanforderungen

Regel: Ab 10.000 EUR ist eine zweite Freigabe erforderlich

@REQ-0417 @Grenzwert
Szenariogrundriss: Status nach erster Freigabe
  Angenommen eine Bestellanforderung über <betrag> EUR wurde erfasst
  Wenn die Kostenstellenverantwortliche die Anforderung freigibt
  Dann hat die Anforderung den Status "<status>"

Beispiele:
  | betrag | status |
  | 9999   | Freigegeben |
  | 10000  | Wartet auf zweite Freigabe |

@REQ-0418
Szenario: Keine Selbstfreigabe
  Angenommen "A. Muster" hat eine Bestellanforderung über 12.000 EUR erfasst
  Und die erste Freigabe ist erfolgt
  Wenn "A. Muster" die zweite Freigabe erteilen will
  Dann wird die Freigabe abgelehnt
  Und der Versuch wird protokolliert

```

Das hypothetische Beispiel zeigt gute Praxis: Die Regel ist benannt, der Grenzwert wird beidseitig geprüft, der Missbrauchsfall ist ein eigenes Szenario, die Sprache ist fachlich statt klickbasiert, und Tags ordnen jedes Szenario einer Anforderung zu. Schon beim Schreiben entstehen Fragen – gilt „ab 10.000 EUR“ inklusive? Diese Klärung ist der eigentliche Nutzen. Für umfangreiche Berechnungslogik sind Entscheidungstabellen oft lesbarer; nicht-funktionale Anforderungen brauchen eigene Prüfverfahren.

Szenarien sind zudem eine Stichprobe, keine vollständige Spezifikation. Zu einem spezifikationsgetriebenen Vorgehen gehören deshalb auch Glossar, Regelkatalog, Schnittstellenverträge und dokumentierte Randbedingungen wie Sicherheit, Performance oder Verfügbarkeit. Alle diese Artefakte werden versioniert, geprüft und gepflegt wie Code.

04 Spec-Driven Development mit KI-Coding-Agenten

Mit KI-Coding-Agenten hat sich eine Arbeitsweise herausgebildet, die ebenfalls als Spec-Driven Development bezeichnet wird: Ziel, Anforderungen, Randbedingungen und Umsetzungsplan werden vor der Generierung in versionierten Dokumenten festgehalten; der Agent arbeitet sie schrittweise ab, Menschen prüfen Spezifikation, Plan und Ergebnis. Die Praxis ist jung, Werkzeuge und Begriffe sind nicht standardisiert, und verallgemeinerbare Aussagen zu Produktivitäts- oder Qualitätseffekten sind mit Vorsicht zu behandeln. Werkzeugunabhängig gelten einige Prinzipien:

- **Spezifikation vor Generierung:** Agenten erhalten geprüfte Regeln, Beispiele und Akzeptanzkriterien statt offener Aufträge.
- **Szenarien als Prüfmaßstab:** Über fachliche Korrektheit entscheidet der Testnachweis gegen vorab vereinbarte Kriterien, nicht der Eindruck.
- **Trennung von Spezifikation und Prüfung:** Erzeugt derselbe Agent Code und zugehörige Tests, können sich Fehlinterpretationen gegenseitig bestätigen. Akzeptanzkriterien verantworten Menschen.
- **Kleine, prüfbare Schritte:** Aufgaben so schneiden, dass jedes Ergebnis mit vertretbarem Aufwand prüfbar ist.
- **Dokumentierte Freigaben:** KI unterstützt, Menschen entscheiden – an definierten, protokollierten Punkten.
- **Evaluation statt Vertrauen:** KI-gestützte Ergebnisse und Agenten selbst brauchen Evaluations- und Testkonzepte – inklusive Prüfung auf Sicherheitsaspekte, unerwünschte Abhängigkeiten und Abweichungen von Architekturvorgaben.

Praktisch bedeutet das: Neben den fachlichen Szenarien erhält ein Agent einen klar umrissenen Kontext – Architekturvorgaben, Konventionen, zulässige Werkzeuge und Berechtigungen. Je präziser dieser Rahmen formuliert ist, desto besser lassen sich Abweichungen erkennen und Ergebnisse im Review bewerten.

Auch beim Entwurf von Szenarien oder beim Aufspüren fehlender Grenzfälle können KI-Werkzeuge helfen. Ihre Vorschläge sind Ausgangspunkt der fachlichen Klärung, kein Ersatz dafür.

05 Traceability und Quality Gates

Nachvollziehbarkeit heißt: Jede Anforderung lässt sich vorwärts zu Szenarien, Tests, Code-Änderungen und Release verfolgen – und jeder Release-Bestandteil rückwärts zu seiner Anforderung. ISO/IEC/IEEE 29148 behandelt Rückverfolgbarkeit als Eigenschaft guter Anforderungen; praktisch entsteht sie über stabile Kennungen in Backlog, Versionsverwaltung, Testmanagement und CI/CD.

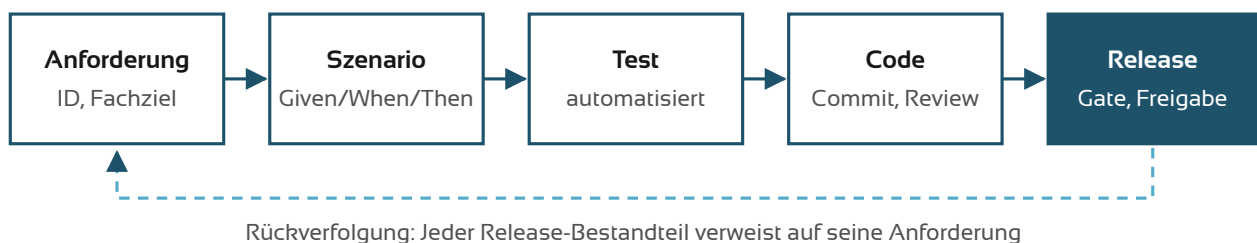


Abbildung 1: Traceability-Kette in einem spezifikationsgetriebenen Vorgehen (schematisch).

Quality Gates sind Prüfpunkte, an denen ein Ergebnis nur bei erfüllten, vorab vereinbarten Kriterien weitergegeben wird – hier ausdrücklich bezogen auf die Spezifikation. Ein Teil lässt sich automatisieren, ein Teil bleibt bewusst menschliche Entscheidung. Gates sollten schlank sein: Ein Gate, das regelmäßig per Ausnahme übersprungen wird, erzeugt nur Scheinsicherheit. ISO/IEC/IEEE 29119 und das ISTQB-Glossar liefern die gemeinsame Terminologie, etwa für Testbasis sowie Eingangs- und Endekriterien.

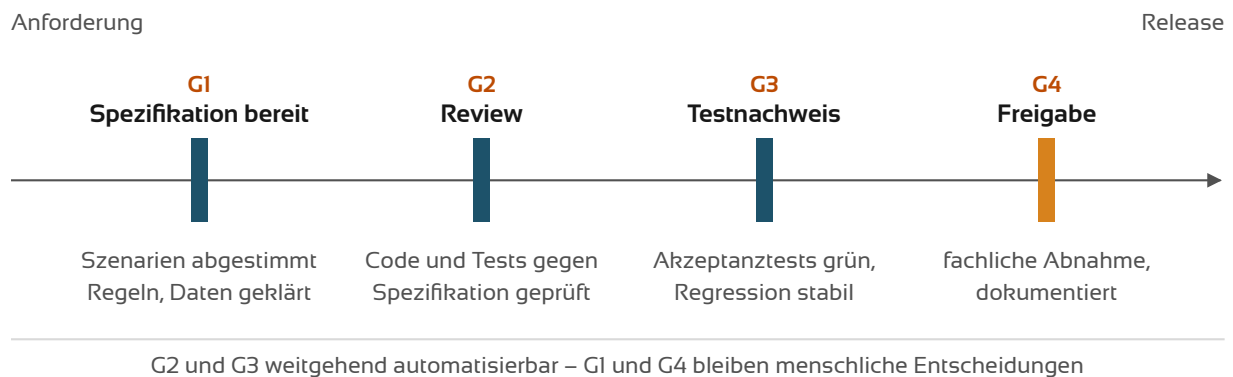


Abbildung 2: Quality Gates entlang der Delivery (schematisch, an das jeweilige Vorhaben anzupassen).

G1 stellt sicher, dass nur abgestimmte und vom Fachbereich freigegebene Spezifikationen in die Umsetzung gehen. G2 prüft Code und Tests gegen diese Spezifikation – unabhängig davon, ob sie von Menschen oder Werkzeugen erstellt wurden. G3 belegt über automatisierte Akzeptanz- und Regressionstests, dass das Verhalten der Spezifikation entspricht. G4 ist die dokumentierte fachliche Abnahme, auf deren Grundlage über das Release entschieden wird.

06 Vorgehen in Phasen

Die Einführung gelingt am besten schrittweise an einem konkreten Vorhaben. Umfang und Tiefe jeder Phase richten sich nach Ausgangslage, Regulierungsgrad und Teamstruktur.

01 Ausgangslage klären

Anforderungs-, Test- und Freigabeprozesse, Werkzeuge, regulatorische Vorgaben und den bestehenden oder geplanten Einsatz von KI-Werkzeugen aufnehmen. Wichtig ist ein ehrlicher Blick darauf, wo Missverständnisse heute tatsächlich entstehen. Ergebnis: Schwachstellenbild und Vorschlag für einen Pilotbereich.

02 Pilot und Spielregeln festlegen

Einen relevanten, überschaubaren Ausschnitt mit klaren Geschäftsregeln wählen; Scope, Erfolgs- und Abnahmekriterien vereinbaren; Sprache, Ablage, Tag-Konventionen und

Freigabeverantwortung festlegen. Ein Pilot mit spürbarem Abstimmungsbedarf macht den Nutzen der Methode für alle Beteiligten sichtbar.

03 Spezifikationen gemeinsam erarbeiten

In kurzen Workshops Regeln, Beispiele und offene Fragen sammeln und in Szenarien überführen; Glossar, Regelkatalog, Schnittstellen und nicht-funktionale Anforderungen ergänzen. Der Fachbereich gibt die Szenarien frei.

04 Automatisierung und Traceability aufsetzen

Szenarien über Step Definitions anbinden und in die CI/CD-Pipeline integrieren, Kennungen durchgängig verknüpfen, reproduzierbare Testdaten und -umgebungen schaffen, Quality Gates konkret definieren. Aus den Testläufen entsteht eine lebende Dokumentation, die jederzeit zeigt, welche Szenarien erfüllt sind.

05 Iterativ umsetzen – auch KI-unterstützt

In kurzen Zyklen gegen freigegebene Szenarien umsetzen. KI-generierte Ergebnisse durchlaufen dieselben Reviews und Gates; neue Erkenntnisse fließen als Änderung der Spezifikation zurück, nicht als stille Anpassung im Code.

06 Auswerten und verankern

Gemeinsam bewerten, wo die Methode Klärung brachte und wo Aufwand; Rollen, Vorlagen, Gates und Governance anpassen und Wissen übergeben, damit die Arbeitsweise nicht an Einzelpersonen hängt.

07 Artefakte und Deliverables

Ein spezifikationsgetriebenes Vorgehen erzeugt keine zusätzliche Dokumentationsschicht, sondern ersetzt verstreute, widersprüchliche Dokumente durch wenige gepflegte Artefakte:

Fachlich

- **Anforderungskatalog** mit stabilen Kennungen, Ziel und Status
- **Feature-Dateien** mit Szenarien, versioniert neben dem Code
- **Regelkatalog** bzw. Entscheidungstabellen
- **Glossar** als gemeinsame Fachsprache
- **Akzeptanzkriterien**, Definition of Ready und Done

Technisch und steuernd

- **Testautomatisierung** mit Step Definitions in der Pipeline
- **Traceability** Anforderung – Szenario – Test – Release
- **Quality-Gate-Definition** mit Kriterien und Verantwortlichen
- **Leitlinien für KI-Werkzeuge** inkl. Review- und Freigabepflichten
- **Freigabeprotokolle** für Abnahmen und Releases

Entscheidend ist die Pflege: Eine Spezifikation, die nicht mehr dem tatsächlichen Verhalten entspricht, schadet mehr als keine. Ausführbare Szenarien haben hier einen strukturellen Vorteil – weicht das System ab, schlägt der Test fehl.

08 Erfolgsfaktoren und Risiken

Erfolgsfaktoren

- **Fachbereich beteiligt:** Szenarien entstehen mit ihm, nicht für ihn.
- **Deklarativer Stil:** Verhalten statt Klickpfade – lesbar und stabil.
- **Klein anfangen:** Ein Pilot überzeugt mehr als eine Großumstellung.
- **Spezifikation als Code:** Versionierung und Review wie bei Quellcode.
- **Klare Verantwortung** für Freigaben, Pflege und Gate-Entscheidungen.

Risiken

- **Gherkin als Testskript:** Nur von QA geschrieben, ohne Klärungsnutzen.
- **Szenario-Inflation:** Zu viele ähnliche Szenarien erhöhen den Pflegeaufwand.
- **Instabile Tests:** Ignorierte Fehlschläge entwerten jedes Gate.
- **Scheinsicherheit durch KI:** Generierte Tests, die generierten Code bestätigen.
- **Vertraulichkeit:** Geschäftsregeln in KI-Werkzeugen nur gemäß Datenschutz- und Sicherheitsvorgaben.

09 Checklist für den Start

VORBEREITUNG UND SPEZIFIKATION

- Pilotbereich mit klaren Geschäftsregeln ausgewählt
- Fachliche Ansprechperson mit Entscheidungsbefugnis benannt
- Erfolgs- und Abnahmekriterien des Piloten vereinbart
- Je Story Regeln, Beispiele und offene Fragen dokumentiert
- Grenzwerte, Fehlerfälle und Berechtigungen als eigene Szenarien
- Jedes Szenario per Kennung einer Anforderung zugeordnet

AUTOMATISIERUNG, KI UND GOVERNANCE

- Szenarien laufen automatisiert in der Pipeline
- Einsatzbereiche für KI-Werkzeuge und Agenten beschrieben
- Agenten arbeiten nur gegen freigegebene Spezifikationen
- Akzeptanzkriterien von Menschen verantwortet
- KI-Ergebnisse durchlaufen dieselben Reviews und Gates
- Freigaben an den Gates dokumentiert

10 Nächster Schritt

Der sinnvollste Einstieg ist klein und konkret: ein Regelbereich aus einem laufenden Vorhaben, ein gemeinsamer Workshop mit Fachbereich, Entwicklung und Qualitätssicherung, am Ende eine Handvoll abgestimmter Szenarien samt offener Fragen. Das zeigt schnell, wie viel Klärungsbedarf in einer scheinbar eindeutigen Anforderung steckt. Auf dieser Grundlage lässt sich fundiert entscheiden, ob und in welchem Umfang ein Pilot sinnvoll ist und welche Quality Gates und Freigaben dafür benötigt werden.

Die dip unterstützt als Beratungs- und Enablement-Partner an der Schnittstelle von Fachbereich und Umsetzung: bei der Einführung von BDD und SDD, beim Requirements Engineering, bei Pilotdefinitionen mit Erfolgs- und Abnahmekriterien, bei Evaluations- und Testkonzepten für KI-gestützte Ergebnisse sowie bei dokumentierten menschlichen Freigaben – damit sich Anforderungen zuverlässig umsetzen, testen und abnehmen lassen, unabhängig davon, wer die Software entwickelt. Kontakt: info@dip-consulting.de.

11 Quellen und Standards

Die folgenden öffentlichen Quellen und Standards bilden den methodischen Hintergrund; alle Beispiele in diesem Whitepaper sind hypothetisch und beschreiben weder reale Kundenvorhaben noch erzielte Ergebnisse.

- Dan North: „Introducing BDD“, Better Software, 2006 – Ursprungstext des Behavior-Driven Development.
- Cucumber: Gherkin Reference – Syntax, Schlüsselwörter und Sprachvarianten von Gherkin.
- Gojko Adzic: *Specification by Example*, Manning, 2011.
- ISO/IEC/IEEE 29148 – Requirements engineering.
- ISO/IEC/IEEE 29119 (Normenreihe) – Software testing.
- ISTQB Certified Tester Foundation Level, Lehrplan und Glossar.

Lassen Sie uns sprechen

Für Infos zu überprüfbaren Spezifikationen, BDD und KI-unterstützter Delivery freuen wir uns auf den Austausch mit Ihnen!

UNTERNEHMEN

DIP Deutsche IT und
Projektberatungsgesellschaft mbH
Rahmannstraße 11 · 65760 Eschborn
info@dip-consulting.de
dip-consulting.de

IHR ANSPRECHPARTNER

Manuel Fernandez

Management Consultant, Geschäftsführer
info@dip-consulting.de

Seit 2013 ist Herr Fernandez Geschäftsführer der Deutschen IT und Projektberatungsgesellschaft und seit 16 Jahren als IT-Berater in vielfältigen Rollen bei strategischen als auch bei implementierungsnahen Projekten im Einsatz.

