Towards a Taxonomy for Applying Behavior-Driven Development (BDD)

David Faragó, QPR Technologies, Karlsruhe
Mario Friske, Freelancer, Berlin
Dehla Sokenou, GEBIT Solutions GmbH, Berlin

BDD - Hype oder Werkzeug der Zukunft?

Was ist BDD?

- Hervorgegangen aus TDD
- Fokus auf Stakeholdern und User-Szenarien
- Universelle, auf Fachlichkeit ausgerichtete Sprache
- Spezifikation semi-formal, mit Step-Implementierung trotzdem ausführbar

BDD - Hype oder Werkzeug der Zukunft?

Das Versprechen:

- Missverständnisse zwischen Stakeholdern treten nicht auf
- Kommunikation durch alle Entwicklungsphasen einheitlich
- Validierung mit Fokus auf Business Value

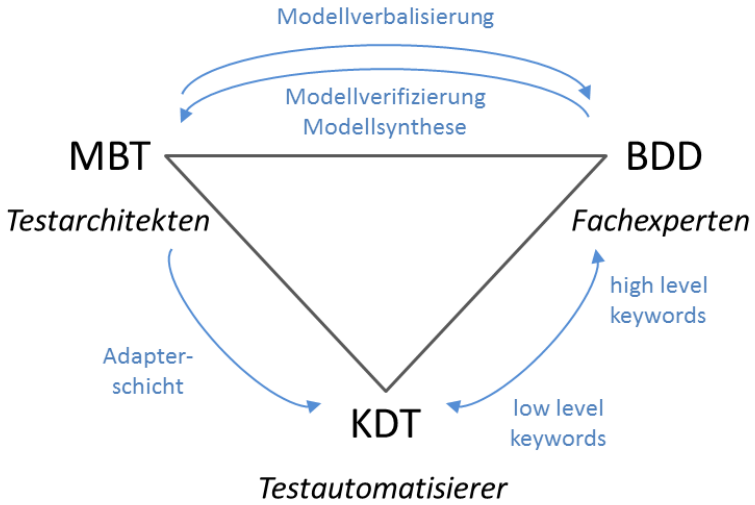
Offene Fragen

Diskussionen im Arbeitskreis TOOP/MBT zu den Themen:

- Wer verwendet BDD und warum?
- Funktioniert BDD über einfache Beispiele hinaus?
- Was sind die Anforderungen an einen erfolgreichen Einsatz?
- Gibt es Vorteile / Nachteile gegenüber anderen Techniken, z.B. MBT?
- Lassen sich verschiedene Techniken mit BDD kombinieren?

Erste Ergebnisse

- Kombination von BDD, KDT, MBT[1)]
- Weitergehende Analyse von BDD
- Sammlung von Vor- und Nachteilen des Einsatzes von BDD



1) Benedikt Eberhardinger, David Faragó, Mario Friske, Dehla Sokenou. *Aktuelle Fragestellungen zum Zusammenspiel von BDD, MBT und KDT.*

Acknowledgments

Wir danken allen Mitgliedern und Diskussionsteilnehmern des AK TOOP/MBT, insbesondere:

Karsten Dörges, Benedikt Eberhardinger, Dierk Ehmke, Matthias Hamburg und Andreas Spillner

Diskussion auf dem letzten AK-Treffen in München bildet Basis für die Taxonomie

Argumente für BDD aus dem AK-Treffen

- nahe an Umgangssprache
- geringe Lern- und Transferaufwände
- logischer Weg um Akzeptanzkriterien zu definieren
- verständlicher als Diagramme
- gute Toolunterstützung
- out-of-the-box in MS-Toolchain
- Eclipse-Support
- eingängiger Workflow
- versionierbar, da textbasiert
- adressiert fehlende Strukturierungsebene

Argumente gegen BDD aus dem AK-Treffen

- nur für Schulbuchbeispiele
- nur flache Szenarien
- lange Szenarien sind schwer wartbar
- Abstraktionsebene nicht vorgeschrieben
- Wiederverwendung auf welcher Ebene?
- Systematische Überdeckung nicht ersichtlich, welche Testbedingungen sind überdeckt?
- Einarbeitungsaufwände, da Open Source
- Workflows und Rollen müssen in großen Projekten definiert werden

Was tun mit den Argumenten?

Sortieren

Gruppieren

Klassifizieren

Weitere Argumente ermitteln

Ziel: Erstellung einer vollständigen Taxonomie

Erste Arbeiten zur Charakterisierung von BDD u.a. in Solis2011[2)] und Holz2015[3)]

Unser Ziel ist die Ausarbeitung einer vollständigen Taxonomie für die Anwendung von BDD

- Analog zur Taxonomie für MBT Approaches[4)]
- Als Entscheidungshilfe für oder gegen BDD

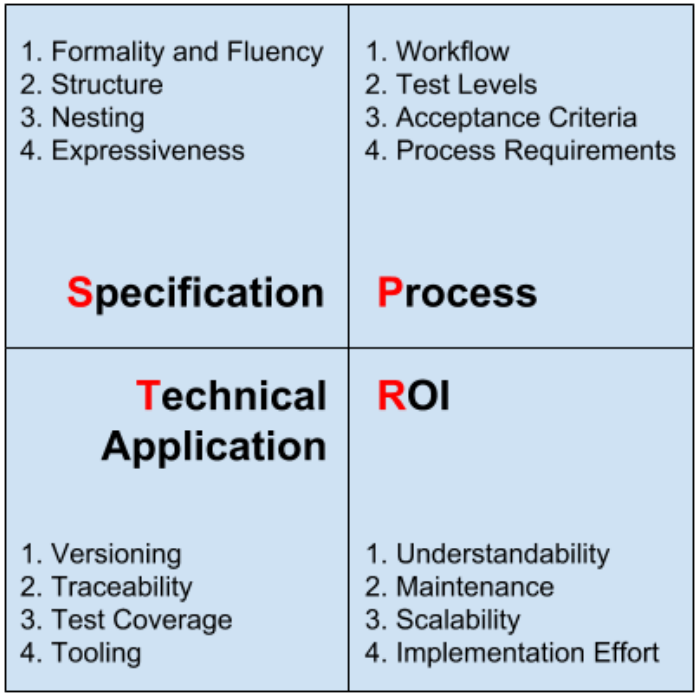
2) Bill Holz. *Increase Collaboration and Drive Agility With Behavior-Driven Development.*
3) Carlos Solís, Xiaofeng Wang. *A Study of the Characteristics of Behaviour Driven Development.*
4) Mark Utting, Bruno Legeard. *Practical Model-Based Testing: A Tools Approach.*

Skizze einer Taxonomie zur Anwendung von BDD

Identifikation und Einordnung der einzelnen Punkte der Diskussion in eine 4x4-Matrix

Aspekte betreffen:

- Specification (S)
- Process (P)
- Technical Application (T)
- Return on Investment (R)



Specification (S)

Kategorie “Specification” betrachtet Struktur, Ausdrucksmächtigkeit und Formalität der Spezifikation in BDD

User Story: Bankkunde hebt Geld ab

Als *Bankkunde,*

möchte ich *vom Geldautomaten Geld abheben*

damit *ich unabhängig von den Filialöffnungszeiten auf mein Guthaben in Form von Bargeld zugreifen kann.*

Szenario: Gewünschter Geldbetrag ist auf dem Konto vorhanden

Gegeben *das Konto verfügt über den gewünschten Betrag*

Und *die Bankkarte ist gültig*

Und *der Bankkunde ist durch PIN authentifiziert*

Und *der Bankautomat enthält das Geld für den gewünschten Betrag*

Wenn *der Kunde den gewünschten Betrag abhebt*

Dann *wird der Betrag ausgezahlt*

Und *das Konto mit dem Betrag belastet*

Und *die Bankkarte zurückgegeben*

Specification: S1 - Formality & Fluency

Verwendung natürlicher Sprache, aber parametrisiert

Nutzung einer semiformalen, an Domäne angelehnten Sprache:

Gherkin (*given, when, then* als Paradigma)

+ Fokus auf Fachlichkeit, nicht Implementierung
+ Leichtere Verständlichkeit auch für Fachexperten
- Szenariobasierte Spezifikation erschwert Wiederverwendung (→ S2)

Specification: S2 - Structure

Wie werden Spezifikationen / Tests strukturiert?

+ Gherkin legt dem Spezifizierer nah, eine Menge von wiederverwendbaren Schritten (Steps) auf Geschäftsebene zu definieren
+ Struktur der Szenarien durch Gherkin vorgegeben
- Keine Vorgabe zur Wiederverwendung auf Automatisierungsebene, Step-Implementierung erfordert deshalb Erfahrung

Specification: S3 - Nesting

Tiefe der Spezifikation, d.h. Verschachtelungsmöglichkeit

Querverweise, d.h. Dokumente zu referenzieren

+ Szenarios sind “self-contained”
+ Outside-In-Ansatz hilft, die richtige Abstraktionsebene zu finden
+ Geschickte Wahl der Abstraktionsebene versteckt Komplexität
○ Überschaubare Anzahl von Ausdrücken
- Geringere Ausdrucksmöglichkeiten (→ S4)

Specification: S4 - Expressiveness

Ausdrucksmächtigkeit der Spezifikation (auch im Vergleich zu anderen Arten von Spezifikationen bzw. Tests, z.B. MBT)

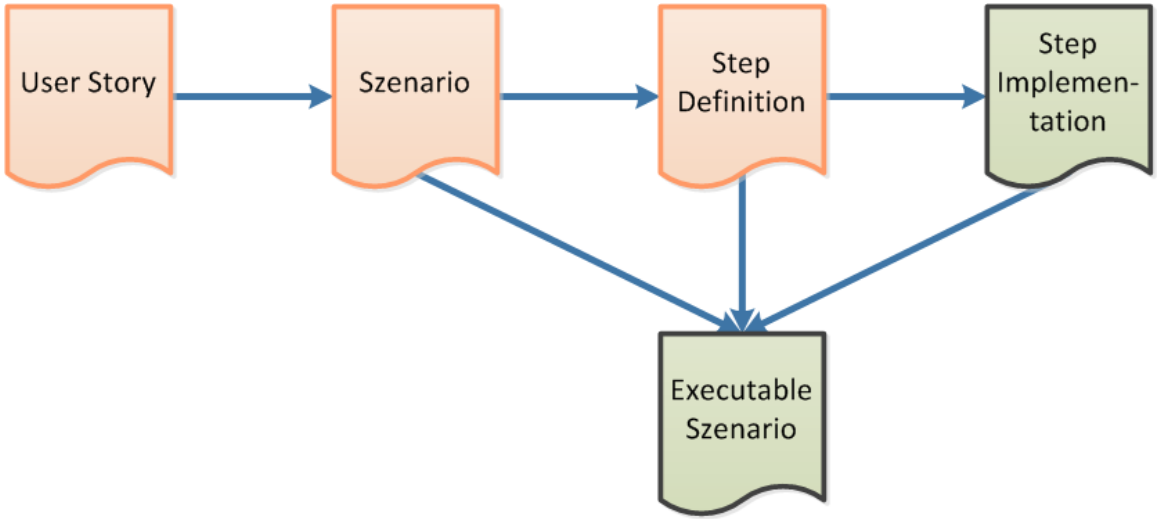
+ Fokus auf signifikante Fälle

- Geringere Ausdrucksmächtigkeit führt zu weniger gefundenen Fehlern
  Aber: Fehler finden ist nicht das primäre Ziel von BDD, sondern die Zusicherung von Eigenschaften

- Inhärente Unvollständigkeit: Fehlende Szenarien werden nicht aufgedeckt

Prozess (P)

Kategorie “Process” betrachtet die Vorgaben, die BDD (ggf.) an den Entwicklungsprozess macht, sowie allgemeine Aspekte



Process: P1 - Workflow

BDD-Workflow (Outside-In) häufig zusammen mit Gherkin als Sprache und Cucumber als Werkzeug (→ P4)

+ Klarer Workflow in Bezug auf Testen

- Weniger klarer Workflow in Bezug auf Entwicklungsseite

- Schwierig vom Unit-Test-zentrierten *Red-Green-Refactor-Cycle* zum höherwertigen *Specify-Develop-Deliver-Cycle* zu kommen
  → BDD wird oft als BDT verwendet

Process: P2 - Test Levels

Szenarios sind user-zentriert auf Akzeptanztestebene

BDD kann auch auf anderen Testebenen verwendet werden, z.B. unter Nutzung der Cucumber-Unterstützung für datengetriebene Tests

+ Theoretisch Unterstützung für verschiedene Testebenen
- Praktisch andere Testebenen nicht methodisch unterstützt
- Wichtige Testebenen fehlen (Stichwort: nichtfunktionale Tests) → hier haben MBT und Fuzzing deutliche Stärken (z.B. bei Security-Tests)

Process: P3 - Acceptance Criteria

Spezifikation ist Akzeptanzkriterium

+ Akzeptanzkriterum wird von Domainexperten und Entwicklern gemeinsam formuliert
+ Ausführbarkeit stellt ständige Qualitätskontrolle sicher
- Outside-In-Ansatz fokussiert Standardfälle, nicht die Fehlerfälle
- Überdeckung nicht garantiert und je nach Szenarien deutlich schlechter als mit anderen Methoden (z.B. MBT)

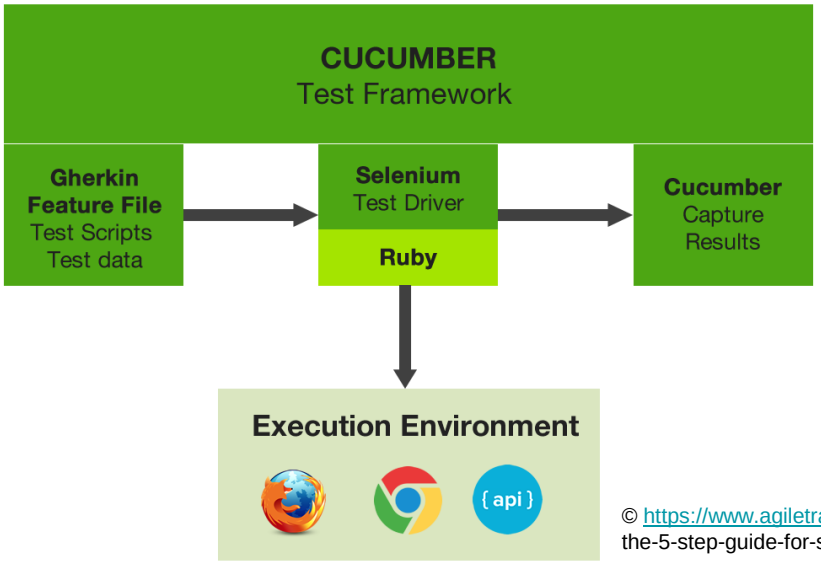
Process: P4 - Process Requirements

Anforderungen an den BDD-Prozess nicht aus anderen Prozessen übertragbar

- Einführung von BDD erfordert ggf. Anpassungen am bestehenden Prozess:
  - Andere Rollen und Verantwortlichkeiten
  - Einbeziehung anderer Stakeholder
  - Andere Werkzeuge

Technical Application (T)

Kategorie “Technical Application” betrachtet technische Aspekte wie Werkzeug-unterstützung und Informationsverwaltung (Versionierung, Traceability).



Technical Application: T1 - Versioning

Wie gut lassen sich die Artefakte versionieren?

+ Textbasierte Artefakte sind leicht versionierbar und vergleichbar
+ Codebasis ist ebenso versionierbar

Technical Application: T2 - Traceability

Rückverfolgung von Tests zu den Requirements, bzw. wenn Test fehlschlägt

Sind alle Requirements durch Tests abgedeckt?

Welcher Code implementiert welches Requirement?

+ Ausführbare Requirements können zum Testen genutzt werden, somit Traceability von Tests zu Requirements und vice versa gegeben
- Zuordnung von Code zu Requirements nur durch entsprechenden Prozess sicherzustellen

Technical Application: T3 - Test Coverage

BDD ist nicht in erster Linie ein Testverfahren, sondern Fokus liegt klar auf Requirements

+ Überdeckung funktionaler Requirements (Szenarien) ist inhärent gegeben
+ Überdeckung auf dieser Ebene auch sehr einfach messbar
- Überdeckung auf anderen Ebenen (Code, nichtfunktionale Requirements, ...) kann nicht garantiert werden
- User-Stories in der Regel nicht vollständig durch Szenarien abgedeckt

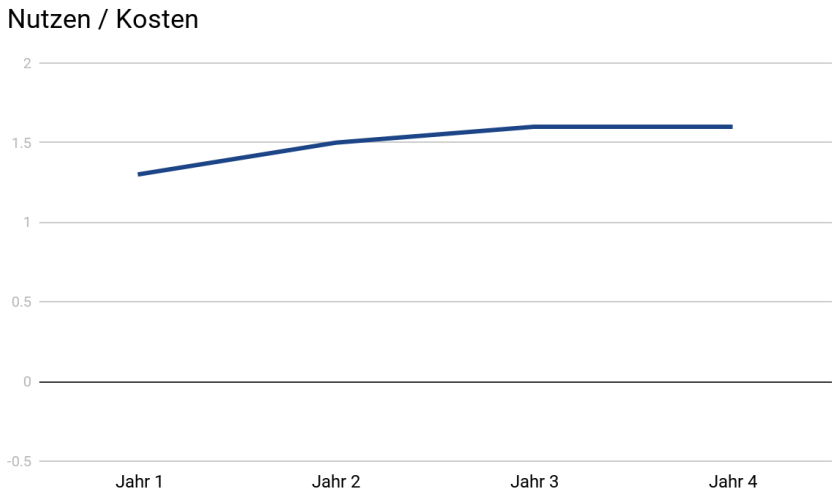
Technical Application: T4 - Tooling

BDD-Tools bieten gute Unterstützung, um Szenarien durch Bindung an SUT ausführbar zu machen

+ breiter Toolsupport (für diverse Programmiersprachen) verfügbar
+ leichte Konvertierung bestehender Automatisierungen
- Aufwand für eigene Step-Implementierung (→ R4)
- Refactoringsupport nur rudimentär vorhanden

ROI (R)

Kategorie “ROI” betrachtet Aspekte, die das Kosten-Nutzen-Verhältnis beim Einsatz von BDD beeinflussen



ROI: R1 - Understandability

BDD-Spezifikationen sind leicht zu lernen und zu verstehen durch

- Semi-Formalismus (→ S1)
- Einfache Strukturierung (→ S2)
- Geringe Tiefe (→ S3)

Weitgehender Open-Source-Toolsupport mit viel Dokumentation/Tutorials

\+ Gemeinsame Sprache für alle Stakeholder = weniger Missverständnisse

\+ Geringe Einstiegskosten, initial wenig Schulungsaufwand

\- Outside-In-Workflow wird manchmal übersehen

ROI: R2 - Maintenance

Komplexe Szenarien auf Requirementseite vs. Komplexität der Step-Implementierung

- Komplexe Szenarien (unabhängig von Seite) schwer zu warten
- Wiederverwendung schwierig, deshalb Tendenz zu Copy&Paste
- Steigender Wartungsaufwand mit Anzahl der Szenarien
    - Pflege nicht nur von Szenarien selbst, sondern auch Step-Implementierung, Tests etc

ROI: R3 - Scalability

Wie gut skaliert BDD in großen Projekten?

- BDD kann gut skalieren, wenn es richtig eingesetzt wird (BDD done right), wie Beispiele erfolgreicher Projekte zeigen[5)]

5) Gojko Adzic. *Specification by Example: How Successful Teams Deliver the Right Software.*

ROI: R4 - Implementation Effort

Gesamtkostenbetrachtung der Einführung von BDD

+ Initiale Kosten niedrig, da weitgehende Toolunterstützung und einfach zu verstehende Konzepte (→ R1)
- Laufende Kosten ggf. hoch, da Wartbarkeit und Skalierbarkeit nur eingeschränkt gegeben (→ R2, R3) – abhängig vom Reifegrad

Zusammenfassung

Von der Diskussion im AK TOOP/MBT zum Schritt in Richtung Taxonomie für die Anwendung von BDD

Vier Kategorien mit jeweils vier Aspekten:

Specification, Process, Technical Application, ROI

Vor- und Nachteile von BDD ermittelt und in Beziehung gesetzt

Situationsabhängig (Projektgröße, Erfahrungen) kann ein Aspekt vorteilhaft oder nachteilig sein

Entscheidungshilfe für Projekte für oder gegen Einsatz von BDD

Diskussion