# Effizientere agile Prozesse – Testfall-basierte Anforderungsdokumentation

**TAD**

Dipl.-Inform. Jörn Koch, Dipl.-Inform. Sebastian Middeke
C1 WPS GmbH
Vogt-Kölln-Str. 30
22527 Hamburg
joern.koch@c1-wps.de
sebastian.middeke@c1-wps.de

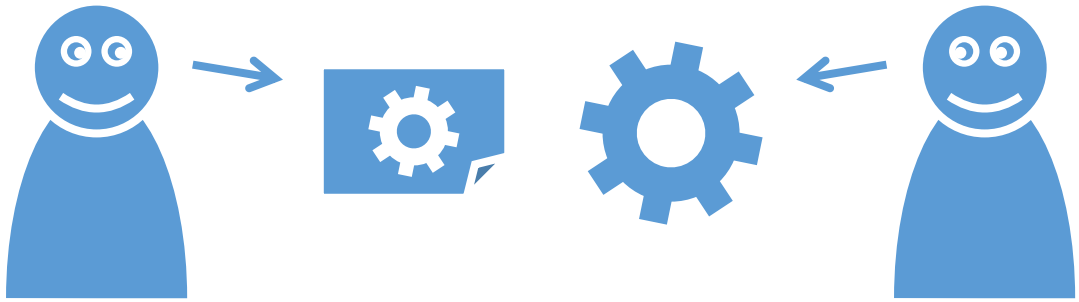
# 1. Motivation - Kunde und Entwickler

- In agilen Projekten arbeiten Kunden und Entwickler gemeinsam an einer Software…
- … und müssen sich verständigen.
- … das tun sie Sprint für Sprint.
- … nach und nach werden alle ein bisschen schlauer.


- … und die Software, die sie gemeinsam bauen, wird immer weiter ausgestaltet.

# 1. Motivation - User Stories und laufende Software

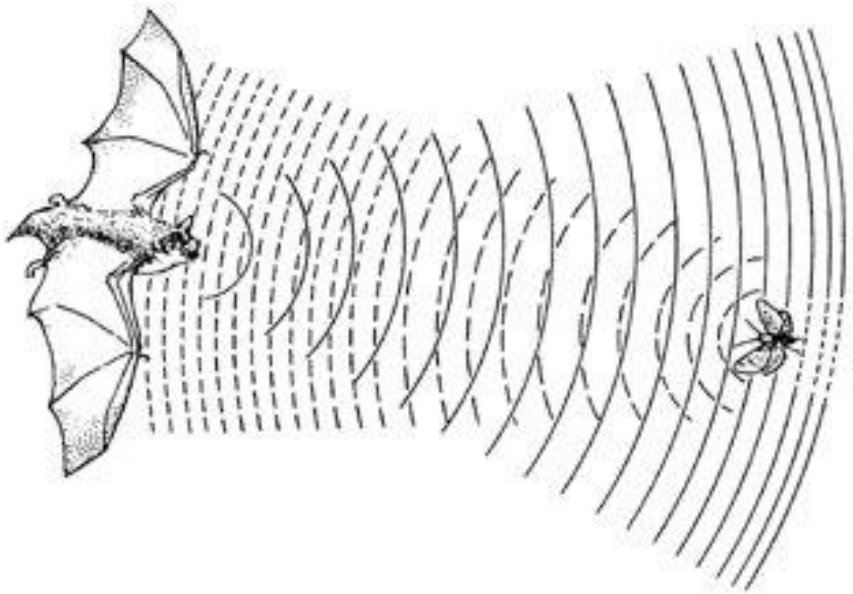
- Die zentralen Artefakte im agilen Vorgehen sind…
- … die User Stories (das SOLL) und
- … die laufende Software (das IST).



- … und jeder schaut auf seine Weise drauf!

# 1. Motivation - Feedback

- Die kontinuierliche, systematische Abstimmung von SOLL und IST ist ein charakteristisches Merkmal des agilen Vorgehens.

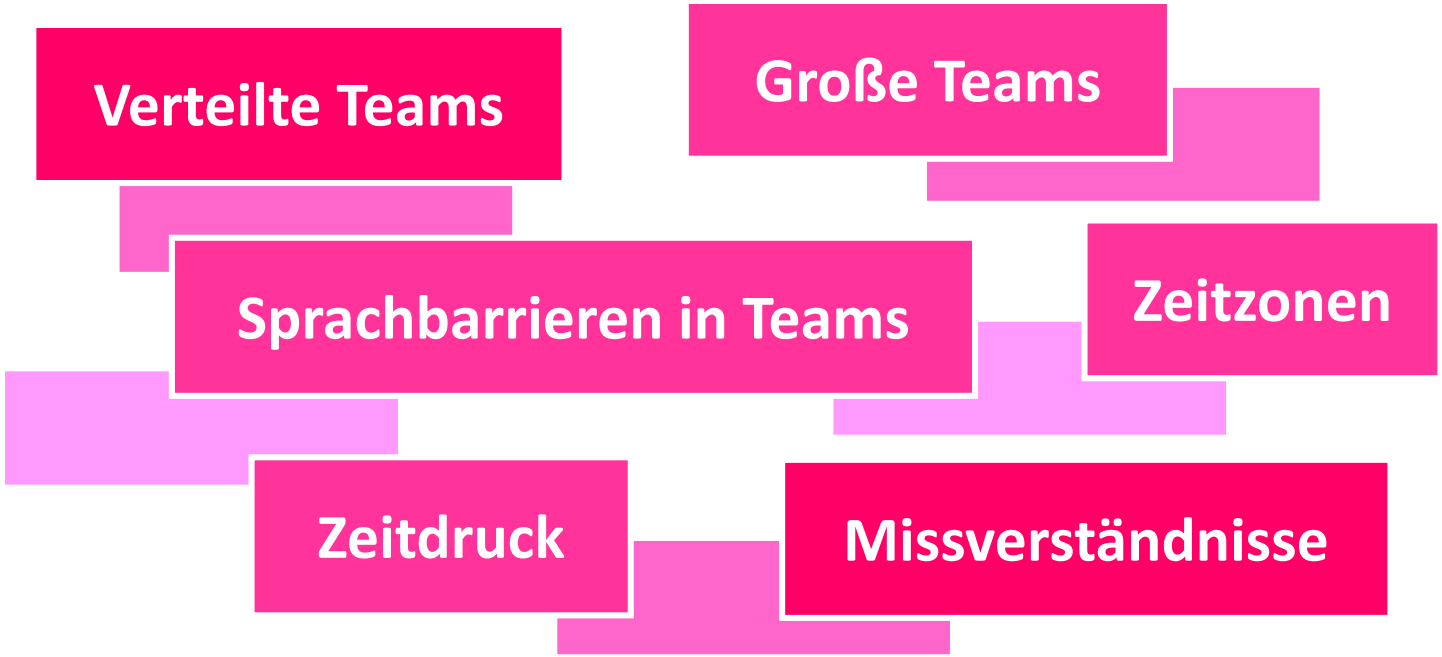


**IT'S ALL ABOUT FEEDBACK!**

(Bildquelle: wissenschaft-online.de)

# 1. Motivation - Kommunikation

- Effiziente Kommunikation ist essenziell für agile Projekte - aber nicht selbstverständlich!



# 1. Motivation - Beispiel: The Ugly Sprint

- Spätestens bei der Abnahme zeigt sich, ob Anforderungen richtig verstanden wurden.


**Warum Kommunikationsprobleme erst NACH der Umsetzung lösen?**

# 2. Der TAD-Ansatz - Die Grundidee

- Wenn Tests am Ende des Sprints helfen, Missverständnisse aufzuklären…
- **… dann helfen sie vorher, Missverständnisse zu vermeiden!**

**Unser Ziel: Gute und unmissverständliche Stories**

# 2. Der TAD-Ansatz - Die Grundidee

- Tests müssen wir am (Sprint/Projekt)-Ende eh spezifizieren, dokumentieren und durchführen.
- **Wenn wir's eh tun müssen, warum dann nicht so früh wie möglich, um möglichst viel davon zu profitieren?**

**Aber nur soviel Doku wie (für den nächsten Sprint) nötig!**

**No Big Design Up Front!**
**Wir wollen agil bleiben.**

# 2. Der TAD-Ansatz - Die Geschichte

- Wir haben die TAD-Idee seit 2005 im Laufe von verschiedenen agilen Projekten erarbeitet und mit der Zeit verfeinert.
- Die verwendete agile Methode in diesen Projekten war durchweg Scrum (mit projektspezifischen Anpassungen).
- Wir glauben, dass der TAD-Ansatz für alle agilen oder auch inkrementellen Methoden geeignet ist. Nicht jedoch für Wasserfall-Methoden.

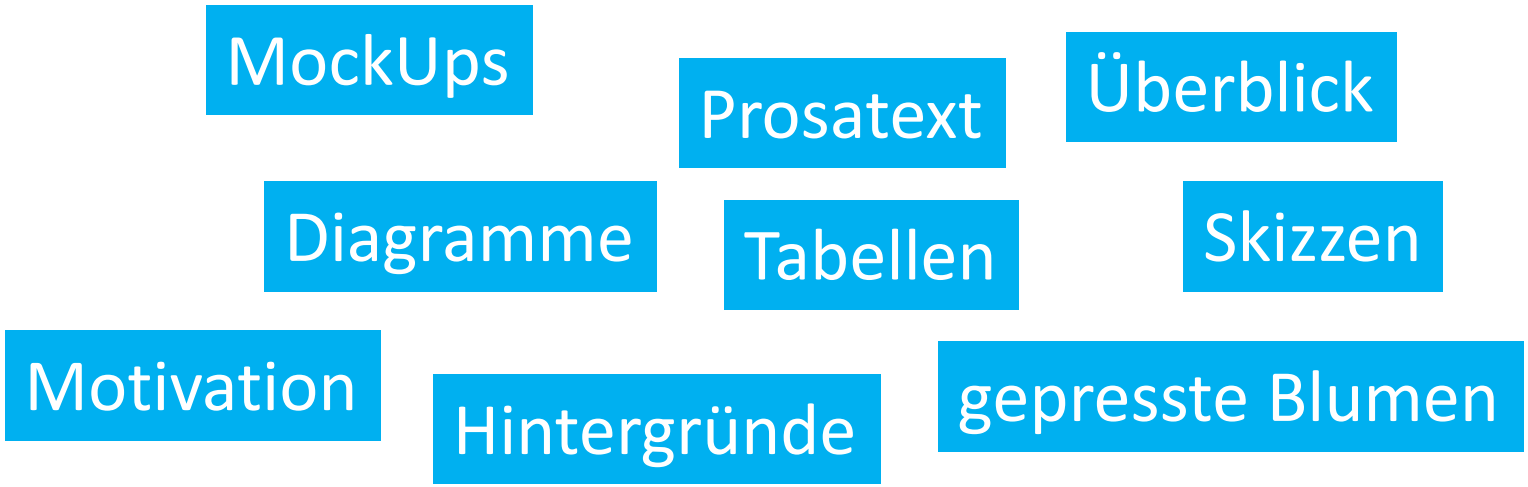
# 2. Der TAD-Ansatz - Form

- Stories bestehen aus relevanten Akzeptanz-Testfällen aus Anwendersicht.

| Stufe/Test | Beschreibung |
|---|---|
| 1. | Anwender bucht eine Rechnung im Zeitraum Ist-Versteuerung ein. Testdaten siehe Tabelle 3: Testdaten für normale Rechnung Ist-Versteuerung. |
| 2. | Anwender ruft den Bericht EÜ/BWA in der Seitenvorschau auf. |
| 2.1. | Der Bericht EÜ/BWA wird in der Seitenvorschau angezeigt. |
| 2.2. | Die Werte im Bericht EÜ/BWA entsprechen der Tabelle 4: Kontrolldaten Einbuchung normale Rechnung Ist-Versteuerung, Nr. 3+4. |
| 3. | Anwender wechselt die Besteuerungsart auf Sollversteuerung. |
| 4. | Anwender ruft den Bericht EÜ/BWA in der Seitenvorschau auf. |
| 4.1. | Der Bericht EÜ/BWA wird in der Seitenvorschau angezeigt. |
| 4.2. | Die Werte im Bericht EÜ/BWA entsprechen der Tabelle 4: Kontrolldaten Einbuchung normale Rechnung Ist-Versteuerung, Nr. 3+4 (keine Änderungen |

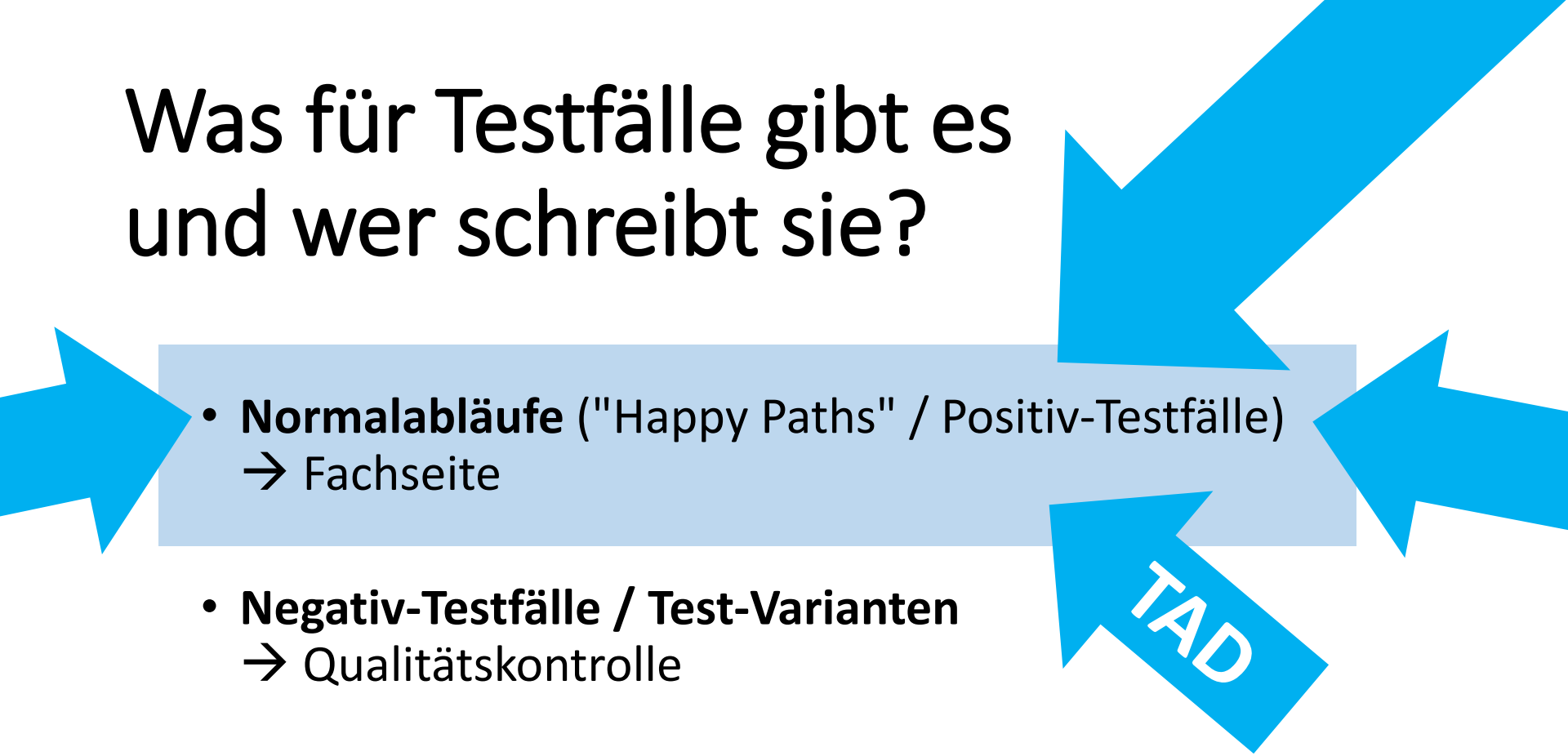
# 2. Der TAD-Ansatz - Prosatexte, Skizzen, Tabellen…?

- Was ist mit dem guten alten Prosatext? "Ich als Anwender in der Rolle … möchte … tun, um …"




- Alles was weiterhilft, ist weiterhin erlaubt!

# Was für Testfälle gibt es und wer schreibt sie?




- **Normalabläufe** ("Happy Paths" / Positiv-Testfälle)
  → Fachseite
- **Negativ-Testfälle / Test-Varianten**
  → Qualitätskontrolle
- **Technische Testfälle**
  → Entwickler
- **Nichtfunktionale Testfälle** (Performance, Usability, …)
  → UX

# Wie finde ich die richtigen Testfälle für eine User Story?

"Was würdest Du als Anwender an der Software ausprobieren, um Dich davon zu überzeugen, dass die User Story nach Deinen Vorstellungen umgesetzt wurde?"

# 2. Der TAD-Ansatz - Testfälle sind...

- **exemplarisch**: Eindeutige Schrittfolge ohne Verzweigungen.
- **relevant:** Prüfen konkrete, wichtige Anforderungen.
- **überprüfbar**: Benutzer-Interaktion plus System-Reaktion.
- **durchführbar**: So detailliert wie nötig. So klar wie möglich.
- **wiederholbar**: Nicht zufällig. Mit Testdaten.
- **einfach:** Keine Umwege. Keine Erklärungen.

# 3. Umgang mit TADs im Entwicklungsprozess

**Fachseite (Entwurf / Spezifikation)**

- TADs geben der Fachseite **Orientierung beim Formulieren** der Anforderungen.
- Anhand der TAD-Testfälle erarbeitet die Fachseite **repräsentative Anwendungsfälle** und geht diese als Gedankenspiel durch.
- Die Fachseite schneidet Anforderungen anhand der TAD-Testfälle in umsetzbare, testbare Portionen: **Zunächst wenige, kurze Testfälle.** In den kommenden Sprints erweiterte, angepasste und zusätzliche Testfälle.

# 3. Umgang mit TADs im Entwicklungsprozess

**Entwickler (Umsetzung)**

- TADs enthalten "naturgemäß" alle für die Umsetzung **notwendigen Details**.
- Anhand der Testfälle verschafft sich der Entwickler **verbindliches Feedback** zur Umsetzung.
- In der Praxis: Der Entwickler fängt mit einem Test an und versucht diesen **zum Laufen zu bekommen**.
- Missverständnisse zeigen sich dem Entwickler unmittelbar durch nicht erfolgreich durchführbare Testfälle.

# 3. Umgang mit TADs im Entwicklungsprozess

**Tester / Testdesigner (Qualitätskontrolle)**

- Die Tester übernehmen die TADs quasi 1:1 in ihre **Testsuite**.
- **Keine Interpretationsfehler!**
- **Keine "Kontrolle der Kontrolle" notwendig.**
- Die Testdesigner **ergänzen Negativtests, Testfall-Varianten, etc.**

# 3. Umgang mit TADs im Entwicklungsprozess

**Fachseite (Abnahme)**

- Die Abnahme kann direkt **auf der Basis der TAD-Testfälle** erfolgen.
- Der **Entwicklungsstand ist anhand der erfolgreichen Tests offensichtlich**.
  Kein umständliches und mitunter schwieriges Zurückführen der Umsetzung auf die Spezifikation.
- **Bugs zeigen sich anhand konkreter Testfälle**, die die Fachseite an die Entwickler übergibt.

# 3. Umgang mit TADs im Entwicklungsprozess

**Fachseite (Änderung von Anforderungen)**

- Anforderungsänderungen **korrespondieren im Umfang** mit den Testfällen.
- Es muss **nur das Minimum an Testfällen angepasst**, umgesetzt und nachgetestet werden.

# 4. Bewertung des TAD-Ansatzes - Kommunikation

**Testfallbeschreibungen sind von der Form her vertraut und für alle Beteiligten gut verständlich.**

**Es entsteht früh ein gemeinsames Verständnis der Fachlichkeit.**

**Diskussionen und Feedback zur Fachlichkeit sind früh möglich.**

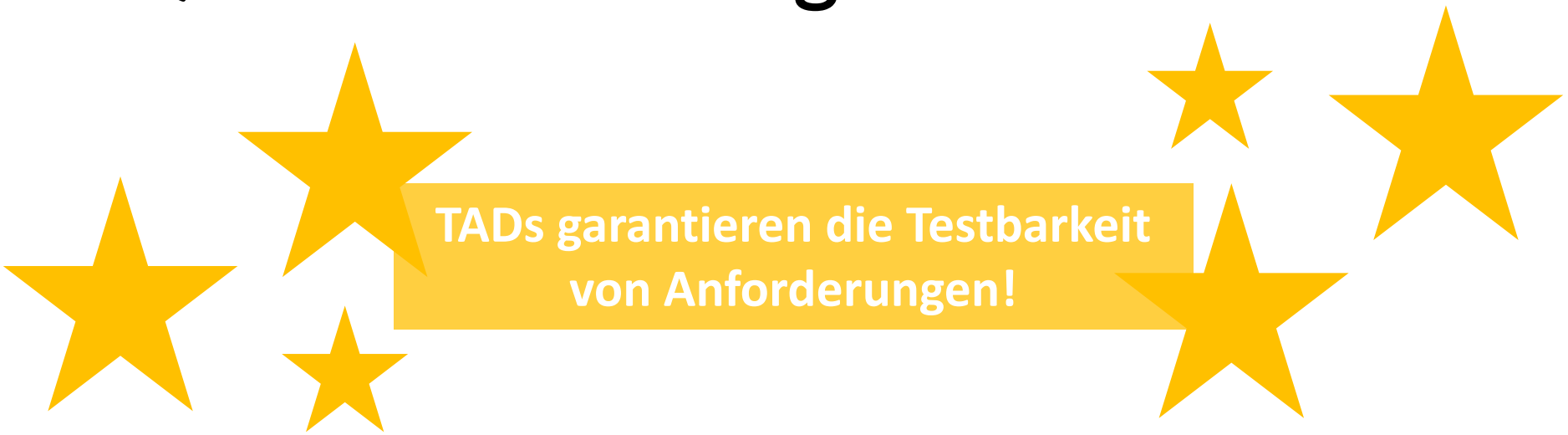
**Jeder kann kompetent sagen, was an der Umsetzung richtig oder falsch ist!**

# 4. Bewertung des TAD-Ansatzes - Detaillierungsgrad

**Testschritte beschreiben nicht nur Leistungs- und Begeisterungs-Features, sondern auch selbstverständliche Basis-Features.**

**TAD-Testfälle enthalten alle für Entwickler relevanten Details bzgl. Interaktionsmöglichkeiten des Anwenders und der Systemreaktion darauf.**

# 4. Bewertung des TAD-Ansatzes - Qualitätssicherung



**TADs garantieren die Testbarkeit von Anforderungen!**

**Die Form der TADs lässt sich gut qualitätssichern, da Mängel meist für jeden offensichtlich sind.**

# 4. Bewertung des TAD-Ansatzes - Testdesign

**Die Autorenschaft der zentralen Testfälle liegt auf der Fachseite.**

**Die wichtigsten Basis-Testfälle und Testdaten sind bereits vorhanden. Geringere Aufwände bei TDD und QK.**

**Es ist keine Interpretation bzgl. der Akzeptanzkriterien nötig.**

# 4. Bewertung des TAD-Ansatzes - QK / Abnahme

**Minimale Aufwände bei der Dokumentation von Bugs.**

**Objektiver und einfacher Abgleich von Spezifikation und Umsetzung! Verbindliche Aussage: Welchen Entwicklungsstand haben wir erreicht?**

**TAD-Testfälle sind Regressionstestfälle.**

# 5. Häufige Kritikpunkte - Extra Aufwand!

**Testfälle schreiben ist anspruchsvoll und aufwändig. Beim Schreiben der Stories fällt nun dieser Extra-Aufwand an.**

**Das Testdesign wird vor die Umsetzung gezogen.
Die Aufwände fallen nicht höher, sondern früher an.**

**In der gelebten Praxis werden Akzeptanzkriterien oft nicht angegeben.**

# 5. Häufige Kritikpunkte - Commitment der Fachseite unrealistisch

**Keine Fachseite lässt sich so frühzeitig festnageln!**

**Damit hätten wir in jedem Projekt ein Problem! Mit TADs merken wir es sehr viel früher.**

**Es klappt i.d.R. besser, wenn die Fachseite nach außen für die Umsetzung verantwortlich ist!**

# 5. Häufige Kritikpunkte - Testfälle werden schnell komplex

**Selbst kleine Features erfordern aufwändige Testfälle! Das steht in keinem Verhältnis!**

**Aufwändige Lösungen sind aufwändig zu testen. TADs machen dies früh spürbar! "Baut mal Altsystem XY neu" ist keine bequeme Option mehr!**

**Einfachheit lohnt sich früh! Für alle!**

# 5. Häufige Kritikpunkte - Testfall-Menge!

**Die schiere Menge der Testfälle wird ab einem Punkt unübersichtlich!**

**Ein geeignetes Testmanagement ist unabhängig vom TAD-Ansatz unerlässlich!**

**Die Menge von Testfällen wächst inkrementell. (Upfront wäre der TAD-Ansatz "unmöglich"!)**

# 5. Häufige Kritikpunkte - Pflegeaufwand der Testfälle

**Ich will neue User Stories eintanken und keine Testfälle pflegen.**

**Es geht jeweils nur um die Testfälle zu einer Story - nicht um den gesamten Bestand an Testfällen!**

**Aus dem Bestand können aber passende Testfälle und Vorlagen ausgewählt und angepasst werden.**

**Testfälle sind änderungsfreundlich und lassen sich in ihren einzelnen Schritten leicht anpassen und um neue Testfälle ergänzen.**

# 6. Ähnliche Ansätze - Stories mit Akzeptanzkriterien

- Stories vor der Umsetzung mit Akzeptanzkriterien zu versehen, ist keine neue Idee.
- Die Form solcher Akzeptanzkriterien bewegt sich dabei von freien Prosaformulierungen bis hin zu formalen Testfall-Spezifikationen (die z.B. durch Tools wie FitNesse ausführbar sind).

**TADs liegen in der Form "dazwischen": In Anwendersprache gehalten und trotzdem eindeutig und präzise und dadurch für alle Beteiligten geeignet.**

# 6. Ähnliche Ansätze - TDD

- Die Idee des TAD-Ansatzes ähnelt der Idee des Test-Driven-Development (TDD): Beginne mit den Tests!

**TDD beginnt erst bei der Umsetzung, TAD bereits davor!**

**TDD bezieht sich auf einzelne Units, TAD auf das Gesamtsystem.**

**TAD ermöglicht verbindliche manuelle Entwicklertests (und damit effizientere Abnahmen)**

# 6. Ähnliche Ansätze - Use Cases

- Nach dem TAD-Ansatz geschriebene User Stories und Use Cases ähneln sich stark:
- Beide sind in der Sprache der Anwender verfasst.
- Beide beinhalten repräsentative Anwendungsfälle.

**TADs sind konkreter als Use Cases, quasi "am System durchführbare Use Cases"**

# "There is no silver bullet" – Aber…

- Die meisten und oft auch schwer zu fassenden Anforderungen sind funktional und basieren auf Benutzerinteraktionen, die sehr gut in Testfällen zu beschreiben sind!

- **TADs helfen, die handwerkliche Seite der Softwareentwicklung effizient und erwartungskonform umzusetzen.**

# Wie führen wir TADs im Projekt ein?

- Tagesworkshops zur Vermittlung der TAD-Struktur und Einbettung der TADs im Prozess.

- Begleitendes Coaching zur praktischen Vermittlung wie TADs geschrieben und verwendet werden.

**Klein anfangen. Auch ein einzelner "Happy-Path"-Test hilft schon sehr viel!**

# FAZIT

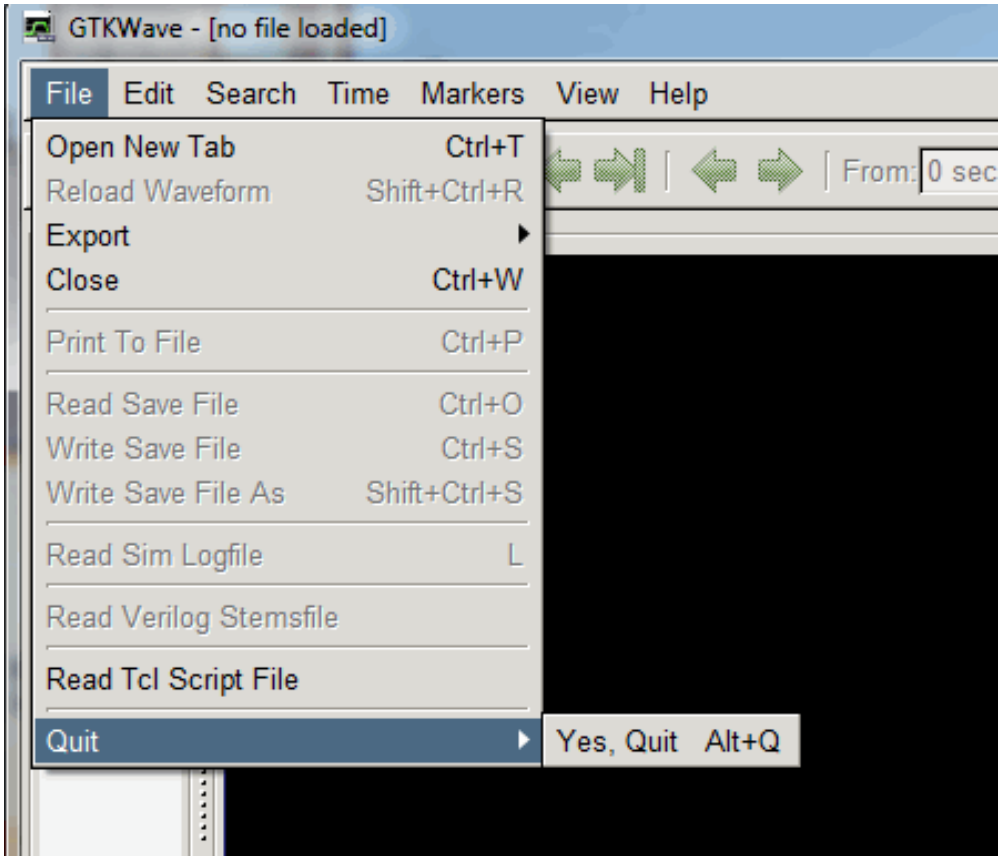
- Eine einfache, bewährte Idee.
- Weniger Fehlentwicklung.
- Verbindliche Entwicklertests und erwartungskonforme Umsetzung.
- Unterm Strich spart der TAD-Ansatz signifikant Aufwände. Gerade in Teams, in denen die Kommunikation schwierig ist oder in großen, unübersichtlichen Projekten, macht sich dieser Effekt besonders bemerkbar.

**Jörn Koch**

jk@c1-wps.de

**Sebastian Middeke**

sm@c1-wps.de




C1 WPS

Vogt-Kölln-Straße 30

22527 Hamburg

Phone +49 40 51 32 26 82

Fax +49 40 51 32 26 83

www.c1-wps.de