# Vorwort

* Quellen sind Hauptsächlich durch Lektüre gestützt
  + Verwendet selbst spezielle Quellen
  + Wer sind diese Quellen?

# ATDD – Kurzbeschreibung

* Vorteile ATDD (S2 – Why ATDD Is Beneficial)
  + Automatisierung bringt weitere Vorteile (Automation After Communication – S5)
* Was ist ATDD (S2 – What Are Acceptance Tests)
  + ATDD sind keine traditionellen Akzeptanztests oder Systemtests
  + Die Implementierung folgt der Erstellung von Anforderungstests
* Wo werden AT’s genutzt (S3 – Where Are Acceptance Tests Used)
  + Keine
* Wer erstellt AT’s (S3 – Who Creates the Acceptance Tests)
  + AT werden gemeinsam von Kunden, Entwicklern und Testern entwickelt (S6/S14)
* Welche Typen der Software werden abgedeckt (S3 – What Types of Software Are Covered)
* Ursprung von ATDD (S5 – ATDD Lineage)
* Zitat: „Acceptance test driven development: The answer is 42. Now implement it.“ (S6)

# Einleitung ATTD

* Software entwickeln (S9/S10)
  + Mit ATTD – ohne ATDD Beispiel / Unterschiede
* Wichtigkeit von ATDD – Beispiel (S10/11)
  + Eine Aussage ungleich ein Test (S11)
    - Eine Aussage kann mehrere AT beinhalten (S14)
  + Absolute Tests sind nicht absolut festgesetzt (S11)
    - 80/20 Prinzip (evtl. Grafik zeigen)
  + AT ist ein Standard für die Verifizierung von richtiger Softwareimplementierung (S12)
* Unterscheidung von AT von anderen Tests (S12)
  + Kunde versteht und spezifiziert mitunter Tests
  + AT ändern sich nicht nach einer Änderung der Implementierung (S14)
* Die Triade
  + Kunde, Entwickler, Tester (S15)
  + Aufgabenverteilung und Agilität (S15)
    - Soccer-Beispiel
  + Weitere Rollen (S16 - Note 2)
  + Erstellen qualitativ hochwertige Software (S21)
* Vergleich zwischen Post- / Pre-Implementation (S17-20)
  + Post – „Klassiche“ / nicht TDD angelehnte Entwicklung
    - Grafik (S17)
    - Nachteile beim Entwickeln
  + Pre – TDD-Style
    - Grafik (S19)
    - Vorteile
      * Quick-Feedback (S18 – Beispiel)
        + Verhindert Loopbacks
      * Face-To-Face (S19)
* ATTD ist „Lean“ und „Agil“ (S20/S21)
  + Von Wem?
  + Ziel
  + Reduziere „Müll“
  + Schaffe Integrität
  + Kontinuierlich verbessern
  + Messbarer Fortschritt

# AT und Test Strategien (S24)

* AT können und sollten in jeglicher Form auftauchen
  + Jeder Teil der Triade deckt einen Testbereich ab
* Testing Matrix (S24 / Xunit Test Patterns: Refactoring Fig 6.1 „Purpose of Tests“)
  + Erkläre 6 Arten (S24)
  + Funktionale Tests sollten häufig auf mehreren Plattformen laufen (S.28)
* Test Pyramide
  + Zusammenhang Test-Matrix und Test-Pyramide
* Deployments / Plattformen (S25/26)
  + Normalfall
  + GothaerSys-Beispiel
* Test Facetten (S. 27)
* Control / Observer Points für besseres Testen (S.27)
* Tests und Anforderungen sind immer Verknüpft (S.28)
* Neuer Test bedeutet neue Anforderung (S27)
  + Neue Anforderung kann ein Detail einer aktuellen Anforderung sein
  + Anforderung ungleich Feature – wird später im Detail erklärt
  + Quote: „Any test created after the code is written is a new requirement or a new detail on an existing requirement.“ (S.27)
    - Wichtige Aussage, da TDD-Style oberste Priorität hat
    - Entwickler tendieren gerne dazu, mal „schnell“ einen zusätzlichen Test zu schreiben 🡪 gegen TDD
  + Ein „Bug“ kann ebenfalls eine neue Anforderung sein (S.27)
    - „Bug“ in diesem Sinne: eine Anforderung, welche durch Missverständnisse oder fehlendes Wissen erst nach der Entwicklung bekannt wurde, und somit kein echter Bug ist, sondern eine neue Anforderung 🡪 hier gilt es stets zu Unterscheiden
      * Analysis bug

# Akzeptanztest Beispiel

* Simple Business-Regel (S29)
  + Beispiel: Erste Tabellen und Interpretation (S30)
  + Austausch innerhalb der Triade face-to-face für richtiges Verständnis von BR
  + Durch Beteiligung mehrerer Stakeholder ergeben sich mehr Fälle (S31)
  + Beispiele der Anforderungen verdeutlichen Anforderungen – schaffen Transparenz über die Anforderung (S37)
    - * Beispiele können direkt als Basis für Tests verwendet werden (S37)
* Einen Akzeptanztest implementieren (S31)
  + Test Script (S31/32)
    - Beispiel: Selenium (Gothaer Beispiel - GEPR)
    - Abhängigkeit vom Interface kann Nachteil sein (Recherchiere: „Nachteil UI Abhängigkeiten“)
  + Test User Interface (S33)
    - Beispiel: GUI / CLI
      * Gothaer Beispiel: Barbara Hamacher – Host-Facettentest?
    - Analogie (S33)
    - Grafik (S34 Figure 4.3)
    - Ruft BR direkt auf (S37)
  + xUnit Test (S34/35)
    - Business Rule Test auf Unit Ebene schwer
      * Änderungen der BR führen zu Änderungen jedes Unit-Tests (S35)
      * Nutzen eignet sich für Testen von implementierter Logik
  + Automatisierter Akzeptanztest (S35/36)
    - Tabellenform muss nicht zwingend konvertiert werden (S35)
      * Weniger Aufwand
      * Möglichkeit: Vorher absprechen, wie Anforderungen notiert werden 🡪 Übereinstimmung mit Style vom Test-Framework für weniger Aufwand und leichtere Umsetzung
    - Kommuniziert direkt mit Modul der BR (S37)
  + Wenn Test nicht als „Test Script“ implementiert ist: Overall Test zusätzlich betrachten
    - Abhängig von Komplexität Anzahl an Testdaten bestimmen (S36)
  + Ein Akzeptanztest verifizieren reicht nicht aus, um sicherzustellen, dass das System den Benutzeranforderungen gerecht wird (Siehe Test Strategien) (S37)

# Entwicklungsprozess

* Beschreibt den Entwicklungsprozess von der Erstellung von Akzeptanztests über den Kunden, bis hin zur Implementierung dieser hinsichtlich von BR’s, verschiedenen Tests auf verschiedener Basis / Ebene
* Grafik: Charter 🡪 Feature 🡪 Story 🡪 Scenario 🡪 Test (S39)
  + Idee: Zeige im Verlauf Grafiken mit einem Aspekt als Fokus, um den Überblick zu behalten, in welchen Schritt man sich befindet
* Charter
  + Definition
  + Erste Idee von Projekt (S40)
    - Vision, Mission, Zielsetzung, Prinzipien (S41)
    - Ziele sollten SMART sein (S40)
      * Specific, Measurable, Achievable, Relevant, Time boxed
      * Beispiele: good / bad practices
    - Projekt-Akzeptanztests (Objectives [Zielsetzung])
      * Management Tests (Note 1: S41)
      * Return of Investment (Recherchieren Internet / Spätere Kapitel)
      * Verzögerung von Projekt 🡪 reduziert ROI (S41)
      * Beispiel (S41)
  + Mini-Charter für jedes Release (Bsp. Gothaer GTRM Zyklus) (S43))
    - Time-Box ist dann oft an Release angelehnt
      * Wenn nicht machbar, dann splitten in mehr objectives (subobjectives)

# Test-Frameworks

* Frameworks (S298)

# Zusammenfassung