

Bachelorarbeit

**Testautomatisierung im Rahmen von Test-Driven-Development**

Fachhochschule Aachen

Medizintechnik und Technomathematik

Scientific Programming

Autor: **Bjarne Herrmann**  
 Leimkaul 4  
 51143, Köln  
 T.: 0176 64484501  
 E-Mail: herrmann.bjarne@outlook.de

Matrikelnummer: 3158998

Gutachter: Prof. Dr. Bodo Kraft  
Zweitgutachter: Stefan Sensler

Abgabetermin: 10.08.2020

Bearbeitungszeit: 6 Wochen

# Sperrvermerk

Die nachfolgende Bachelorarbeit enthält vertrauliche Daten und Informationen der Gothaer Systems GmbH. Veröffentlichungen, Vervielfältigungen oder die Weitergabe des Inhalts der Arbeit im Gesamten oder in Teilen sowie das Anfertigen von Kopien oder Abschriften (auch in digitaler Form) sind grundsätzlich untersagt. Ausnahmen bedürfen der schriftlichen Genehmigung der Gothaer Systems GmbH.

Die Bachelorarbeit ist nur den Korrektoren sowie den Mitgliedern des Prüfungsausschusses zugänglich zu machen.

Die Weitergabe oder Veröffentlichung ist nicht erlaubt bis zum 01.08.2070.

# Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die Bachelorarbeit mit dem Thema

**Testautomatisierung im Rahmen von Test-Driven-Development**

Selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Name: Bjarne Herrmann

Köln, den 20. Juli 2020

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Unterschrift

# ATDD – Vorwort

Dieses Kapitel dient als Einblick in Acceptance Test-Driven Development (kurz: ATDD), um einen Überblick zu erhalten. Im späteren Verlauf werden kommende Themen im Detail aufgegriffen und erläutert.

Im Rahmen der stetig komplexer werdenden Softwareentwicklung rückt ATDD immer weiter in den Vordergrund. Besonders durch ständige Anpassungen im agilen Kontext, sich ändernden Anforderungen und veränderbaren Technologien muss die Qualität von Software gewährleistet werden. ATDD gewährleistet eine kontinuierliche Absicherung oben genannter Punkte. Zudem ist es wichtig zu verstehen, dass nicht nur die Erstellung von Tests bei der Entstehung von Software, sondern das Beibehalten dieser mit möglichst wenig zukünftigen Anpassungen für die Automatisierung eine hohe Relevanz besitzen. Akzeptanztests in diesem Kontext grenzen sich von den traditionellen Benutzer-Akzeptanztests[[1]](#footnote-1) oder Systemtests[[2]](#footnote-2) ab. Die Implementierung von Anforderungen folgt stets der Erstellung von Akzeptanztests. (Pugh, 2010 S. 2) Sie fungieren als Blackbox-Tests[[3]](#footnote-3), die unabhängig von der Implementierung entstehen. Weiterhin gilt: Der Zeitpunkt und die Art und Weise der Erstellung dieser Tests differenziert diese von den klassischen Tests. (Pugh, 2010 S. 3) Während klassische Tests durch die Implementierung definiert werden, gilt dies bei Akzeptanztests in ATDD nicht. Da die Absicht des Tests selbst zu priorisieren ist, gibt es auch keine Regel der festen Eingliederung in eine Testebene[[4]](#footnote-4). Solche Tests lassen sich beispielhaft als Validierungstests für Eingaben, oder ebenso als Designverifizierungstests, die eine Unit oder Komponente testet, nutzen. In beiden Fällen gilt es zu gewährleisten, dass der Akzeptanztest sicherstellt, dass eine bestimmte Anforderung des Kunden im System „akzeptiert“ ist. (Pugh, 2010 S. 3) In der klassischen Softwareentwicklung werden Tests in der Regel von Entwicklern oder Testern[[5]](#footnote-5) erstellt, um eine Kundenanforderung – meist nach der Implementierung dieser – auf dessen Gültigkeit zu prüfen. Akzeptanztests innerhalb des ATDD-Kontexts werden aber gemeinsam vom Kunden, Entwicklern und Testern entwickelt, um ein gemeinsames Verständnis über diverse Anforderungen und dessen Details zu erhalten (Pugh, 2010 S. 3,14).

Während ATDD bereits lange bekannt war, wurde dies im Rahmen von „extreme programming“[[6]](#footnote-6) wiederbelebt. Dabei beinhaltet das Manifest diverse Entwicklervorgehen:

* Acceptance Test-Driven Development (kurz: ATDD)
* Example Driven Development von Brian Marick (kurz: EDD)
* Behavior Driven Development von Dan North (kurz: BDD)
* Story Test-Driven Development von Joshua Kerievsky (kurz: SDD)
* Domain-Driven Development von von Eric Evans (kurz: DDD)
* Executable Acceptance Test-Driven Development (EATDD)

Jedes dieser vorgehen besitzt das Ziel von Erstellung qualitativ hochwertiger Software, indem Entwickler und Tester mit spezifischen Methoden unterstützt werden, den Kunden und dessen Bedürfnisse zu verstehen.

ATDD kann mitunter Methodiken aus anderen Entwicklungsphilosophien verwenden, muss dies aber nicht zwingend. Dazu gehören zum Beispiel das Konzept der Ubiquitous Language[[7]](#footnote-7) aus DDD sowie das Given-When-Then Template[[8]](#footnote-8) und die Tabellendarstellung von Tests aus BDD. (Pugh, 2010 S. 5,6) Vorher genannte Punkte werden im späteren Verlauf konkretisiert.

# Literaturverzeichnis

**Pugh, Ken. 2010.** *Lean-Agile Acceptance Test-Driven Development.* (UTB Schlüsselkompetenzen, 3218). USA : Pearson Education, Inc., 2010. 978-0-321-71408-4.

1. Akzeptanztest in der klassischen Entwicklung (auch Abnahmetests genannt) dienen zur Überprüfung, ob die Anforderung erfolgreich implementiert worden ist. Dabei gilt es lediglich um die Erfüllung eines Tests, nicht um die Art und Weise der Abnahme und Erstellung dieser. [↑](#footnote-ref-1)
2. Tests, die in der Regel überprüfen, ob das System die Anforderungen erfüllt und richtig arbeitet. [↑](#footnote-ref-2)
3. Tests, dessen zu prüfenden Implementierungen nicht bekannt sind. [↑](#footnote-ref-3)
4. Testebene, im Sinne der vier Teststufen: Unittests, Integrationstests, Systemtests, Abnahmetests [↑](#footnote-ref-4)
5. Eine spezielle Entwicklerrolle, die auf Softwaretesten spezialisiert ist. [↑](#footnote-ref-5)
6. Entwicklermethode, in der das Lösen von Programmieraufgaben im Vordergrund steht und Anforderungen des Kunden in kleinen Schritten annähert. [↑](#footnote-ref-6)
7. Eine Kommunikation mit dem Ziel, einheitliche Begrifflichkeiten zu verwenden und ein gegenseitiges Verständnis zu gewährleisten. [↑](#footnote-ref-7)
8. Ein Format für die Erstellung von Tests in einer formalen Sprache, die für Techniker und Kunden verständlich ist. [↑](#footnote-ref-8)