DevBoost

Acceptance Test-Driven Development

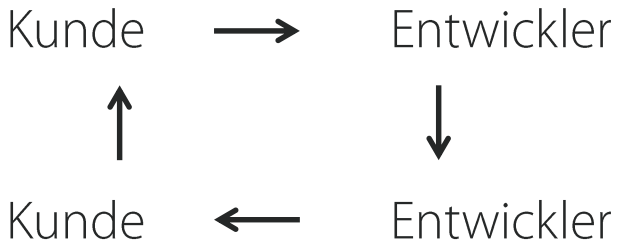
- Ausführbare Tests in natürlicher Sprache schreiben -

Mirko Seifert, DevBoost GmbH

18. April 2013, JUG Saxony, Dresden

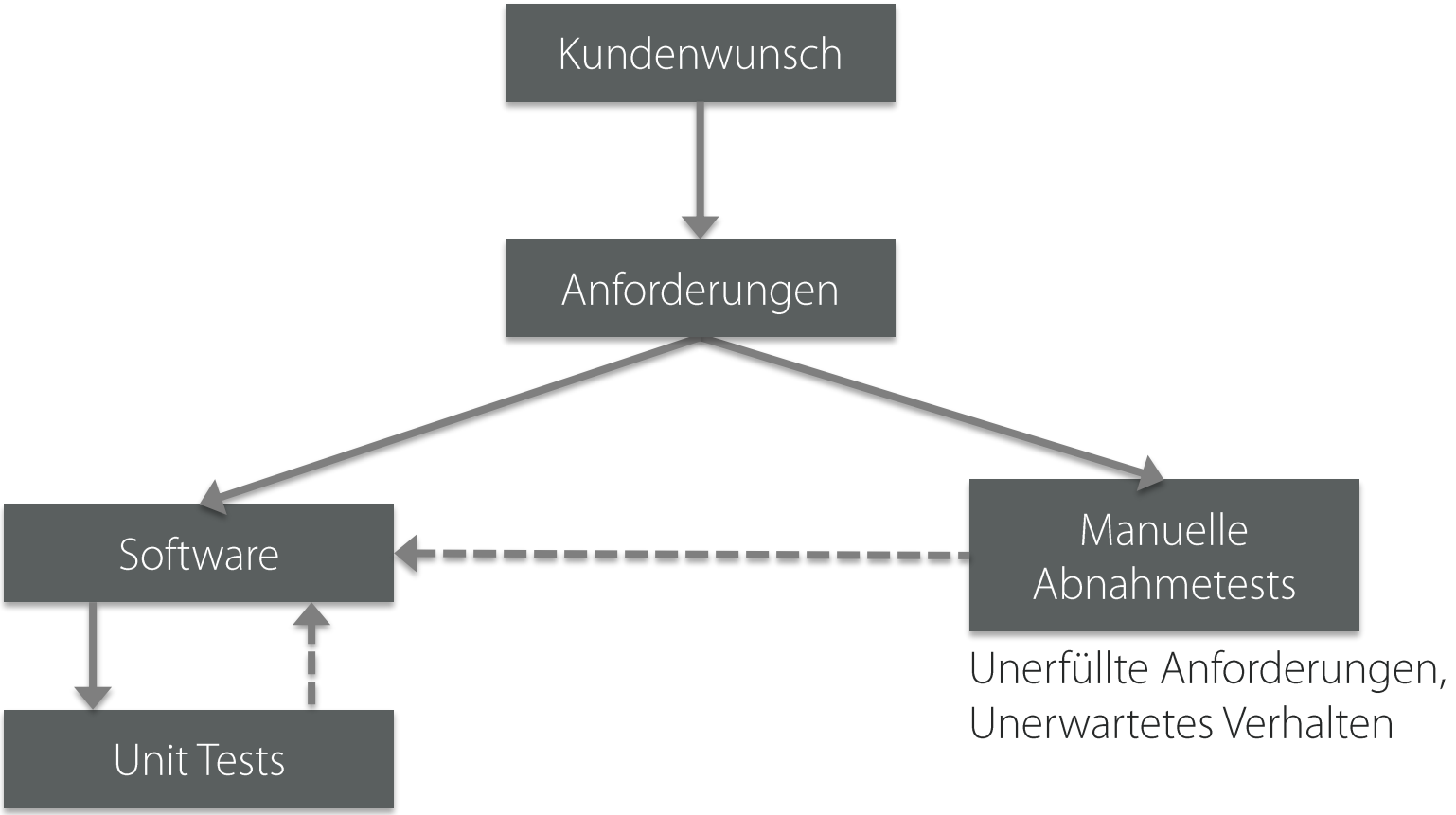
Ausgangssituation: Softwareentwicklung

- Softwareentwickler bauen Software, um die Anforderungen ihrer Kunden zu erfüllen (Job to be done)
- Kommunikationsproblem in der Anforderungsanalyse



- Anforderung als wichtiges Akzeptanzkriterium bei der Auslieferung

Klassisches Vorgehen



Probleme

- Verständnisprobleme zw. Kunden und Entwicklern werden zu spät erkannt
- Anforderungsspezifikation ist "totes" Artefakt
- Communication Gap zwischen
  - Anforderungsspezifikation und Softwareimplementierung
  - Anforderungsspezifikation und Abnahmetest
- Asynchronität zwischen Implementierung und Akzeptanztest
- Unit Tests validieren aus Entwicklersicht, nicht aus Kundensicht

Lösungsansatz: Wir entwickeln agil

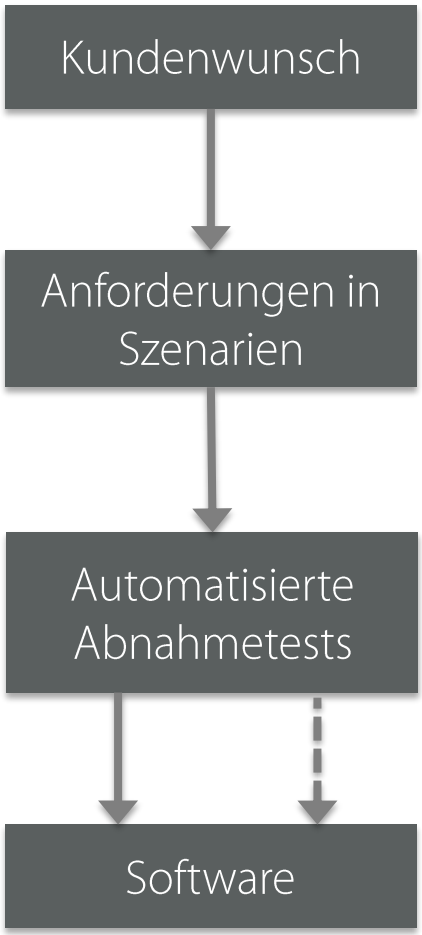
- Viele, kurze Iterationen sichern eine kontinuierliche Annäherung von Kundenbedürfnis und Software

Lösungsansatz: Wir entwickeln agil

Ja, klasse Idee! Aber:

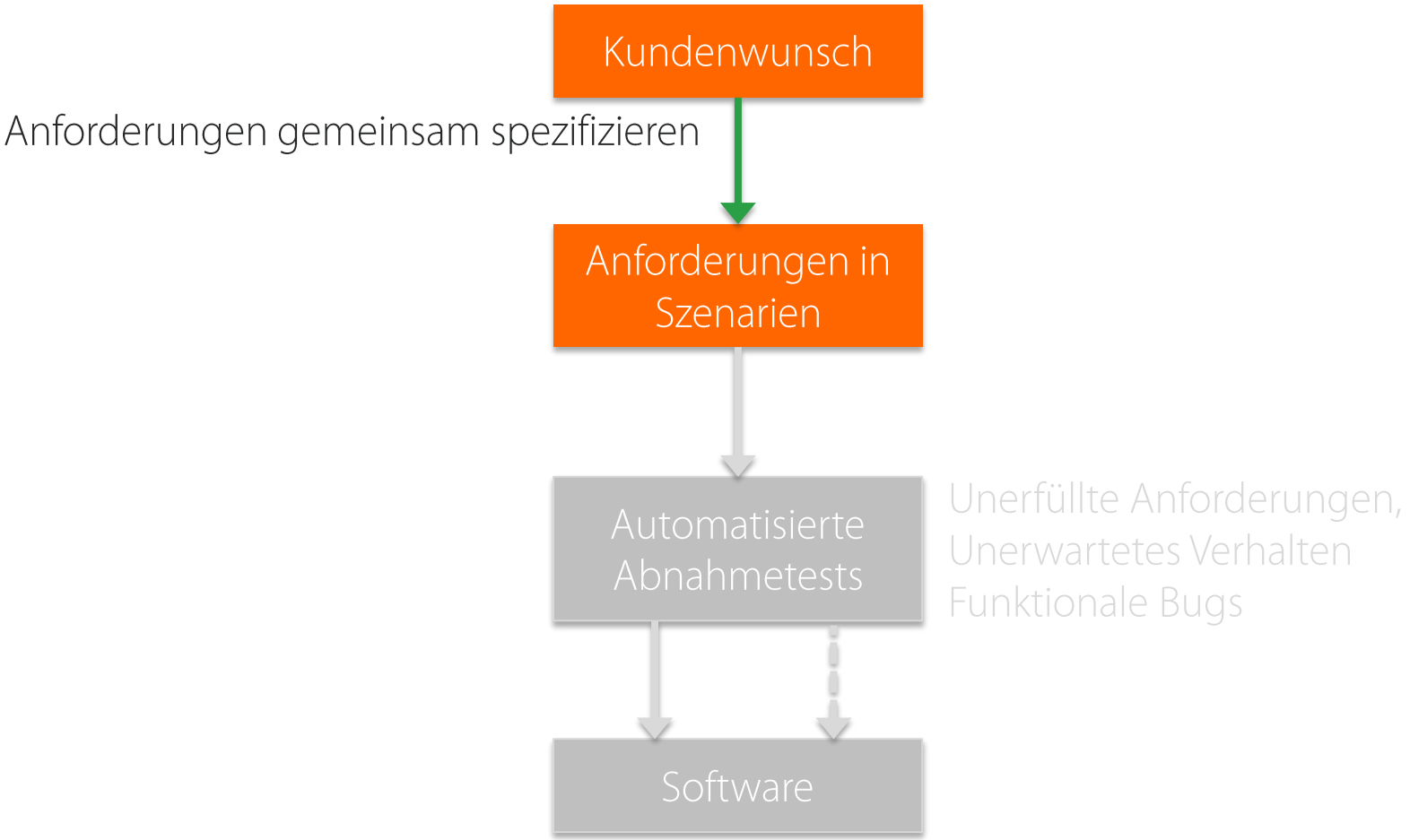
- Agilität erhöht die Geschwindigkeit, verträgt sich das mit Testen?
- Sind umfangreiche Anforderungsspezifikationen nicht unnötiger Aufwand?
- Wie soll ich bei kurzen Iterationen die Anforderungen, die Software und die Tests warten?

Acceptance Test Driven Development (ATDD)



Unerfüllte Anforderungen,
Unerwartetes Verhalten,
Funktionale Bugs

Acceptance Test Driven Development (ATDD)



Anforderungen gemeinsam spezifizieren

- Anforderungsspezifikation erfordert Zusammenarbeit von Kunden, Analysten, Entwicklern und Testern
- Kunden definieren Was sie benötigen und Warum
- Entwicklungsteam gestaltet Wie Anforderungen umgesetzt werden

- Natürliche Sprache ist das Mittel der Wahl für allseitiges Verständnis
- Anwenderszenarien mit realistischen Bespielen begünstigen Verständigung

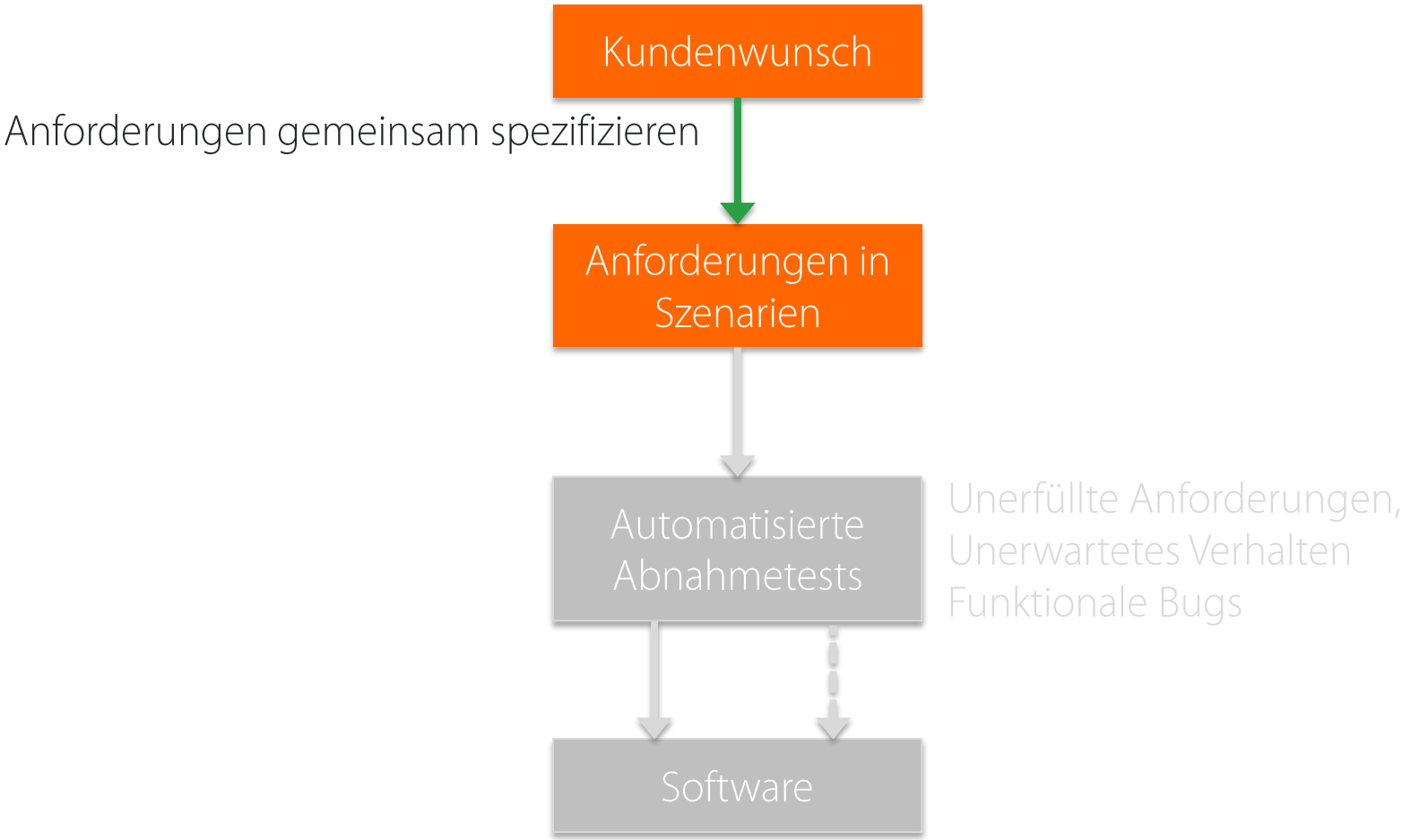
Anforderungen gemeinsam spezifizieren

Beispiel Flugbuchungssystem (Airplanes, Flights, Passengers)

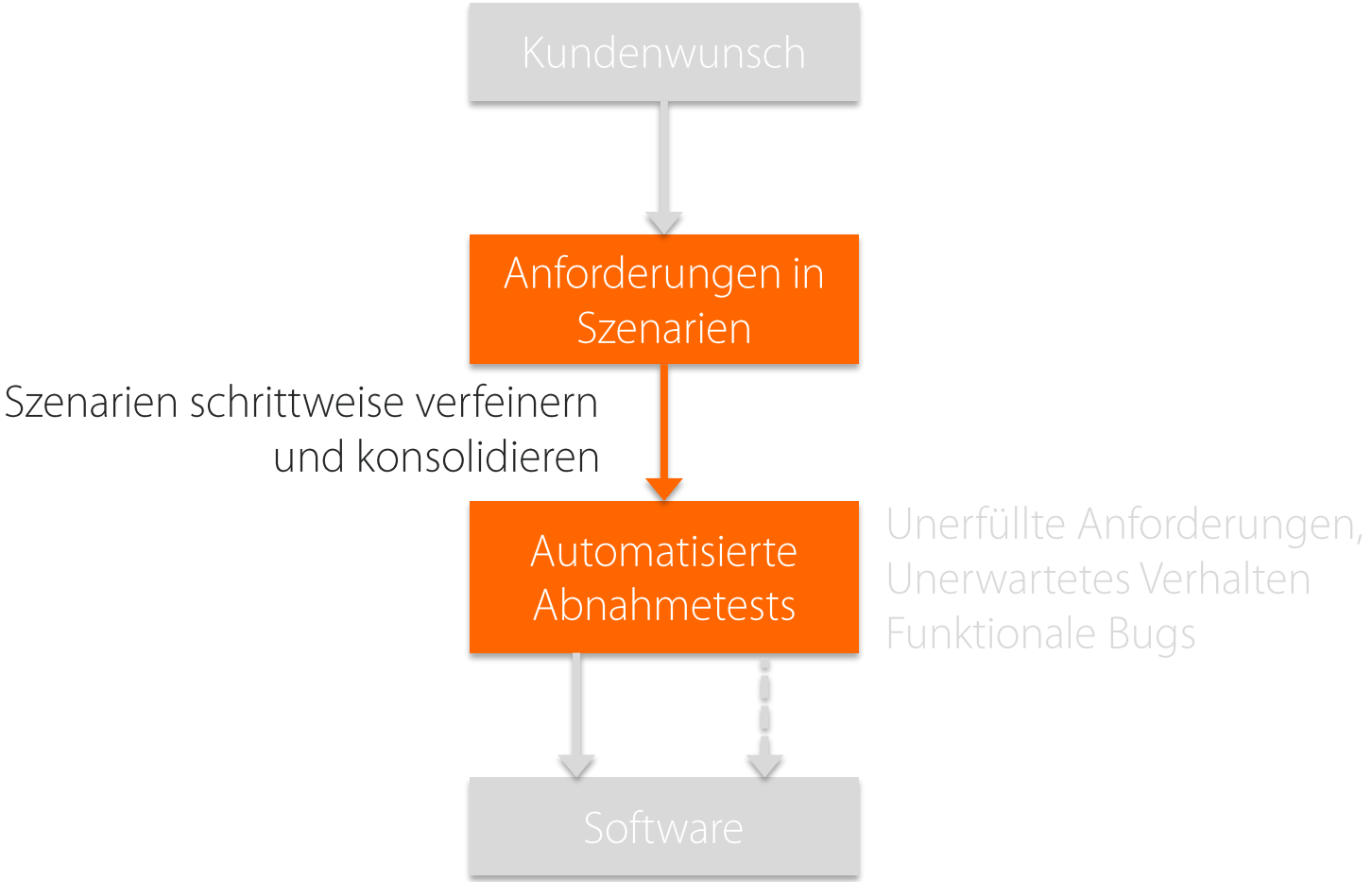
Story: Booking a Flight

Given a Passenger John Doe Given an Airplane Boing-787 With 200 total seats Given a flight LH-1234 With 200 free seats	Eingabe
Book seat for John Doe at LH-1234	Systemfunktion
Assume Success	Ergebniserwartung

Acceptance Test Driven Development (ATDD)



Acceptance Test Driven Development (ATDD)



Szenarien schrittweise verfeinern und konsolidieren

- Verwendete Sprache konsolidieren (Redundanzen entfernen, Muster finden und konsistent verwenden, Terme vereinheitlichen)
- Mit initialen Szenarien und Daten experimentieren, Ausnahmefälle durchspielen und in Szenarien verwandeln

- Die ursprünglichen Beispiele weiterentwickeln
- Natürliche Sprache zur Abstimmung der Verfeinerungen immer noch notwendig

Szenarien schrittweise verfeinern und konsolidieren

Beispiel Ausnahmefall In welchem Fall soll eine Buchung fehlschlagen?

Story: Booking a Flight with Passenger already booked

Given a Passenger John Doe

Given an Airplane Boing-787

Given a flight LH-1234

With 200 free seats

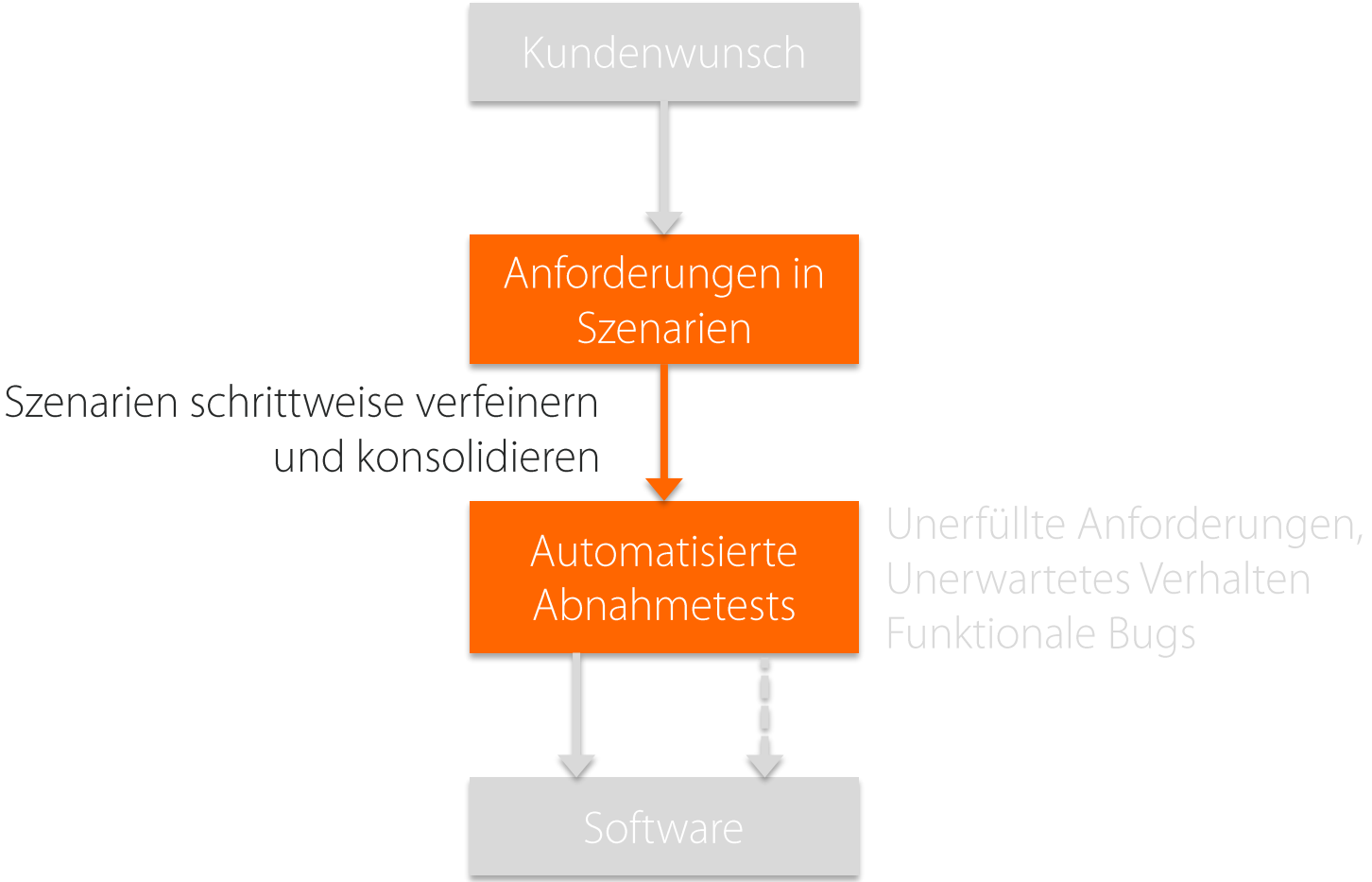
Book seat for John Doe at LH-1234

Assume Success

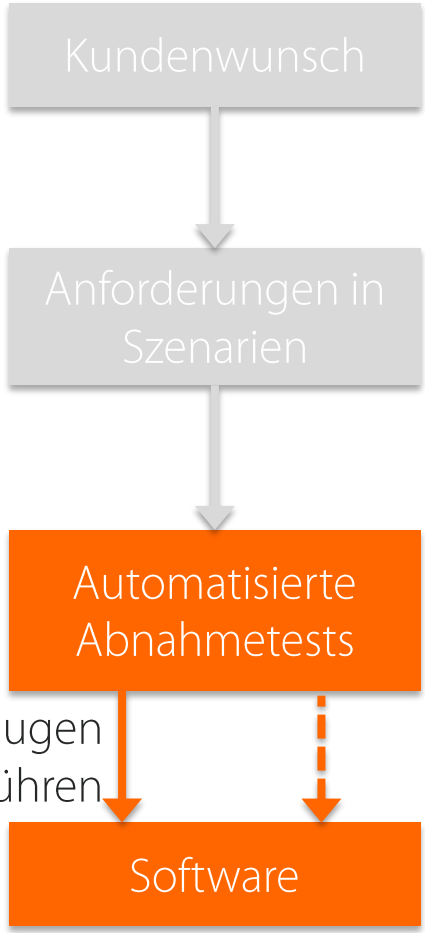
Book seat for John Doe at LH-1234

Assume Failure

Acceptance Test Driven Development (ATDD)



Acceptance Test Driven Development (ATDD)



Unerfüllte Anforderungen,
Unerwartetes Verhalten
Funktionale Bugs

Ausführbare Tests automatisiert erzeugen und wiederholt ausführen

Ausführbare Tests automatisiert erzeugen…
…und wiederholt ausführen

- Nur testbare (ausführbare) Szenarios bleiben lebendig und sind wertvoll
- Testerzeugung muss direkt aus Szenarios erfolgen und darf diese nicht verändern
- Testerzeugung muss für Entwickler einfach und flexibel umsetzbar sein
- Testerzeugung muss auf bestehende Test-Frameworks aufsetzen (JUnit, TestNG, HtmlUnit, jMock, EasyMock, Arquillian, etc)
- Erzeugung von Testfällen aus Szenarios sollte sich nahtlos in das Entwicklungsvorgehen integrieren
- Testausführung sollte schnell Feedback erzeugen
- Tests müssen auf dem CI System regelmäßig laufen

Tools, Tools, Tools

Jnario

NatSpec

Vergleich

Jnario

Mapping von Text auf Steps Definitions	Annotation von Sätzen mit Code	Mapping von Sätzen auf Test Support Methoden
Kommandozeilen-orientiert	Eclipse Integration	Eclipse Integration
Interpretiert	Generiert Test Fälle	Generiert Code
Gherkin	Gherkin, Specs (Xtend)	Frei
ATDD	ATDD	ATDD + Lean Modeling
Community	Individual	Professional

Vorteile von ATDD

- Verbesserte Kommunikation und Zusammenarbeit im Team und mit Kunden
- Fokussierung auf die wesentlichen und kundenrelevanten Softwarefunktionen
- Direkte Messung des aktuellen Projektfortschrittes
- Höhere Entwicklungseffizienz durch klare, kontinuierliche Verständigung, weniger Missverständnisse, weniger Iterationen und weniger "tote" Spezifikationen
- Höhere Produktqualität durch frühestmögliche, kontinuierliche Überprüfung des Implementierungszustandes bzgl. der Akzeptanzkriterien

Die nächsten Schritte

Tools ausprobieren

- Cucumber: http://cukes.info/
- Jnario: http://jnario.org/
- NatSpec: www.nat-spec.com - Kostenlose 30-Tage Trial Version

Literatur

- Specification by Example: How Successful Teams Deliver the Right Software (Gojko Adzic - Manning)
- The Cucumber Book: Behaviour-Driven Development for Testers and Developers (Matt Wynne, Aslak Hellesoy - Pragmatic Programmers)
- ATDD in der Praxis: Eine praktische Einführung in die akzeptanztestgetriebene Softwareentwicklung (Markus Gärtner - dpunkt.verlag)
- ATDD by Example (Markus Gärtner - Pearson Education)

Danke! Fragen?

http://www.devboost.de

mirko.seifert@devboost.de