

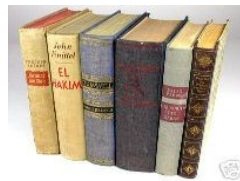
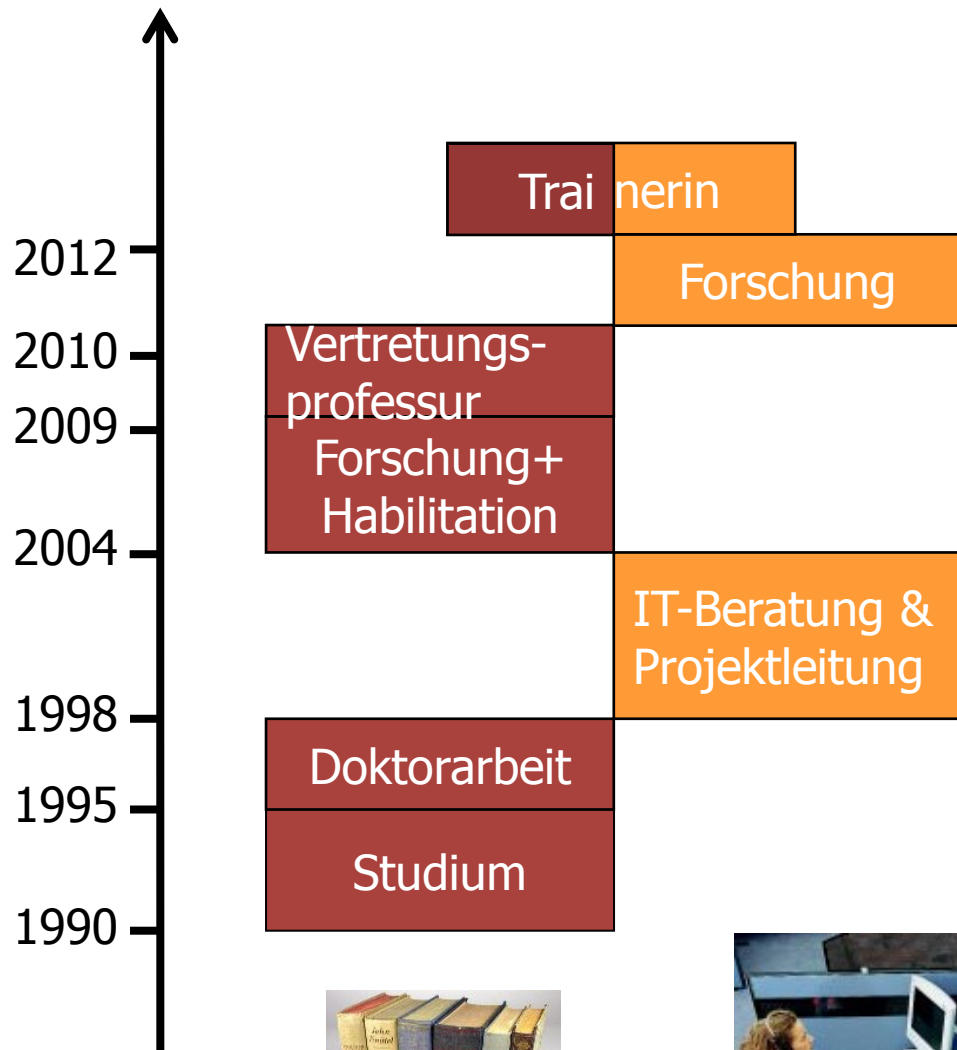
Modellbasiertes manuelles Testen: Techniken und Tücken

16.04.2015 Stuttgarter Testtage

Dr. Andrea Herrmann

Freiberufliche Trainerin für Software Engineering

herrmann@herrmann-ehrlich.de



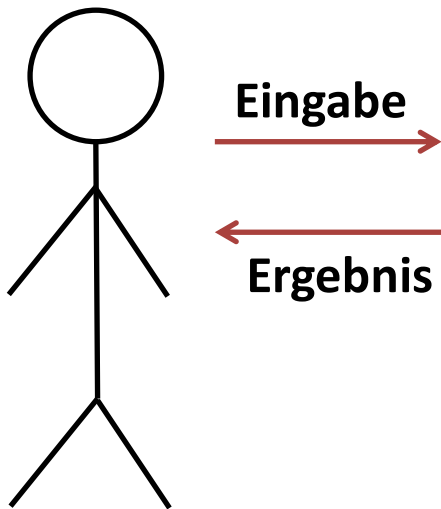
Überblick des Vortrags

- Ansätze für händisches modellbasiertes Testen
- Warum eine Automatisierung praktisch schwierig ist
- Ergebnisse eines Experiments: Fehler beim Testfall-Entwurf

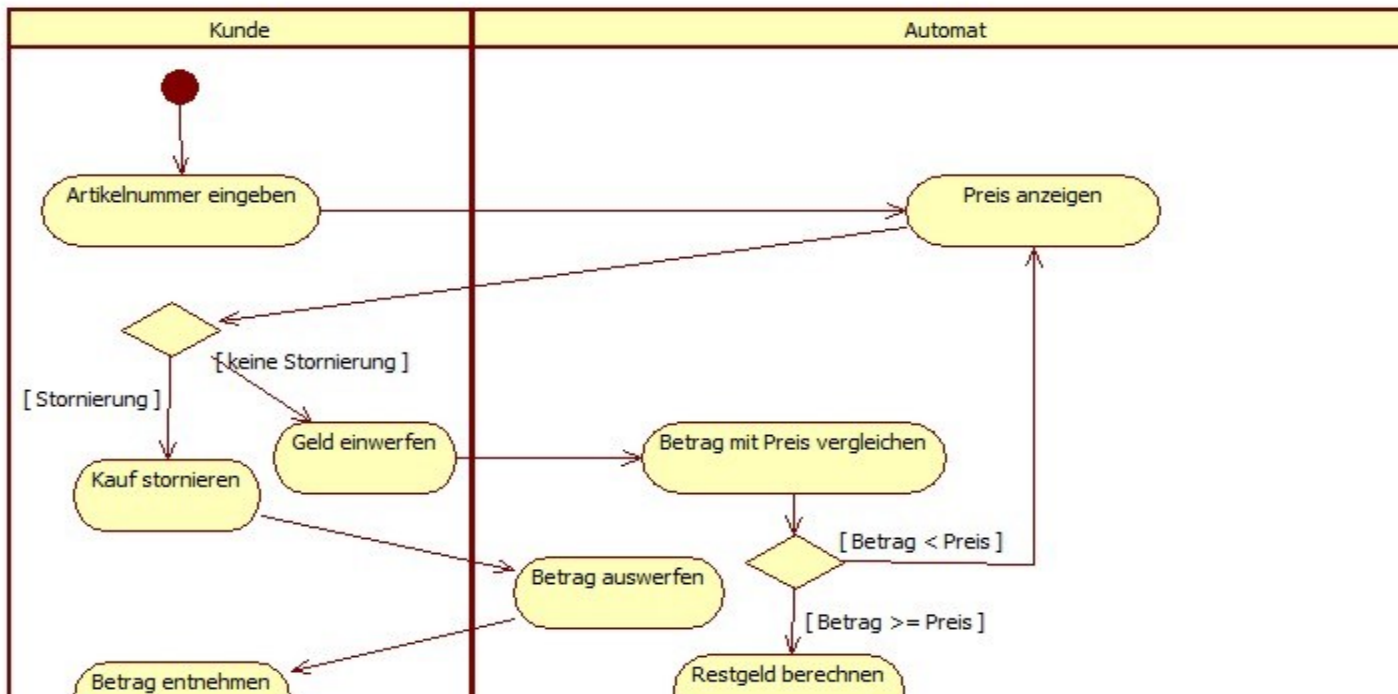
Getränkeautomat



Getränkeautomat



Motivation



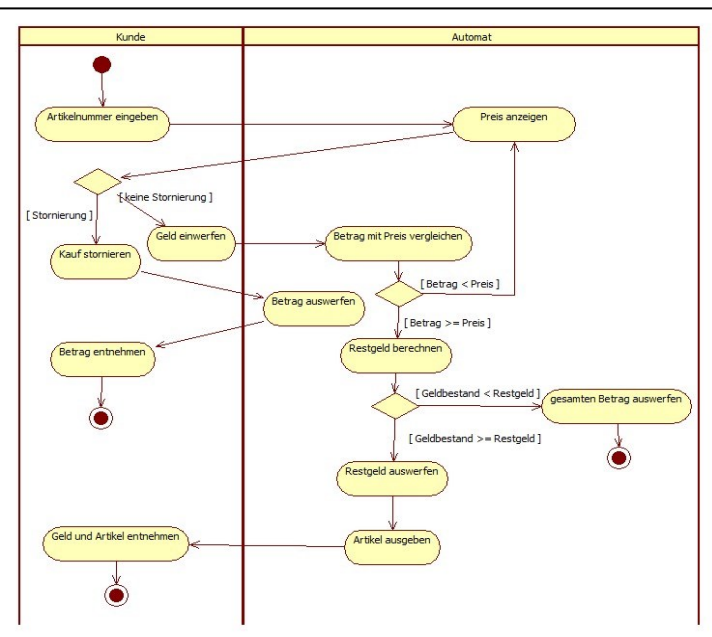
Testfall 1 Name: Stornieren

Vorbedingung: keine

Nr.	Testschritt	Eingabedaten	Soll-Ergebnis
1	Artikelnummer eingeben	gültige Artikelnummer	
2	Preis anzeigen		Preisanzeige
3	Kauf stornieren	Stornierung	
4	Betrag auswerfen		Betrag ausgeworfen
5	Betrag entnehmen		

Motivation für modellbasiertes Testen

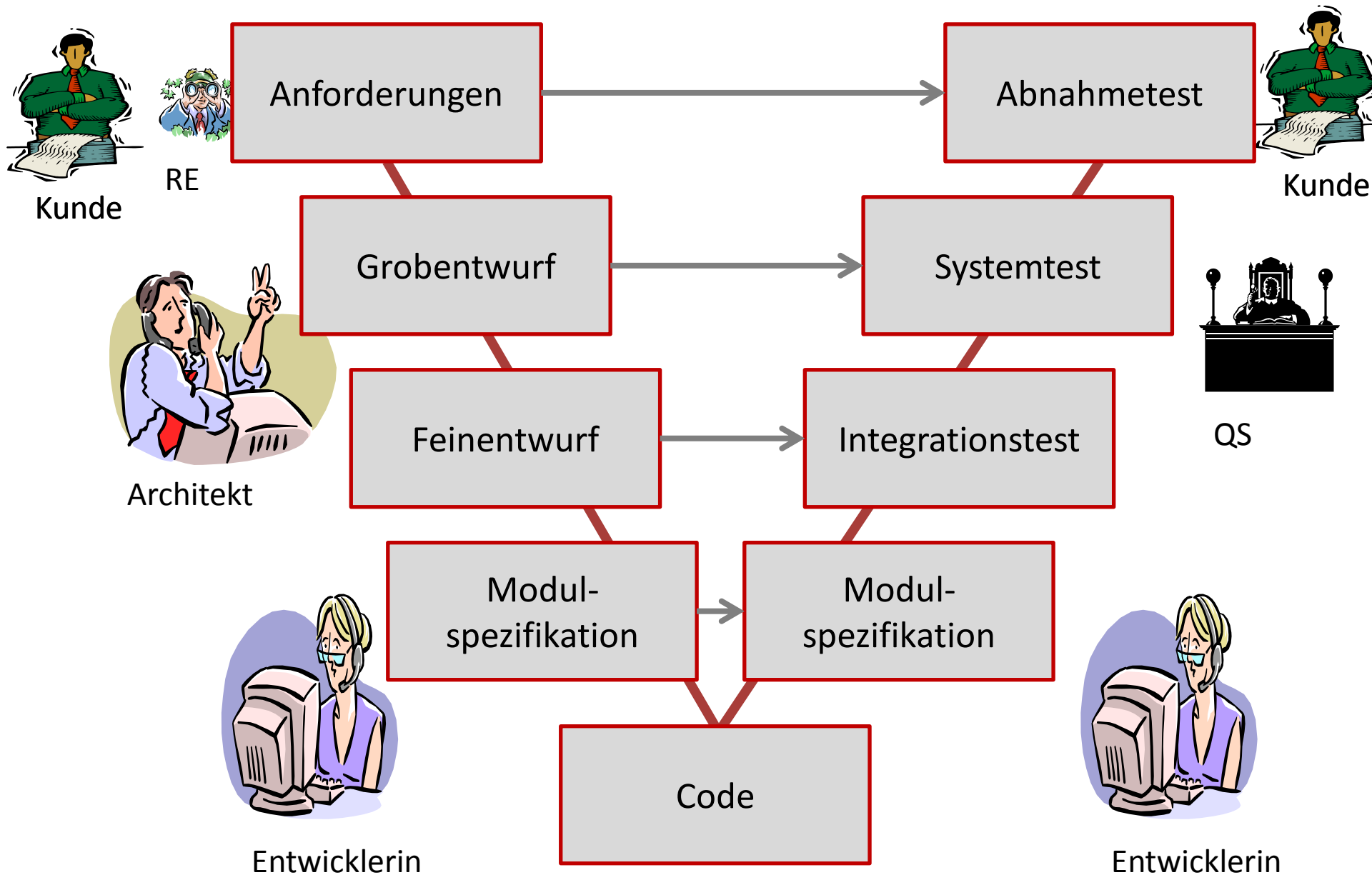
- 1) Anforderungen = Grundlage für Entwicklung und Test
- 2) Modellbasierte Tests finden mehr Fehler
- 3) Nur testbare Anforderungen sind gute Anforderungen



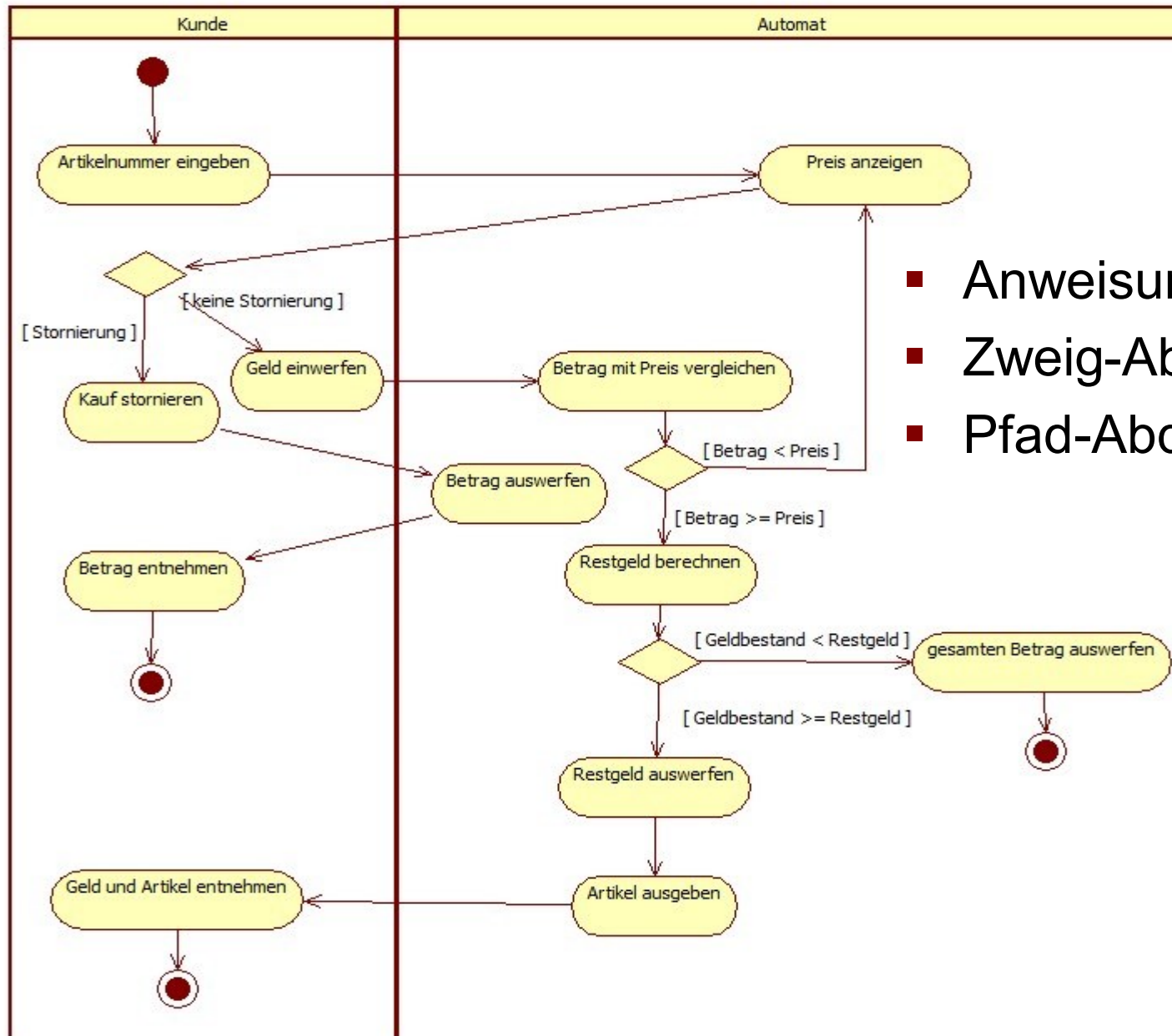
Test-fall 1			
Vorbedingung:		Name: Stornieren	
Nr.	Testschritt	Eingabedaten	Soll-Ergebnis
1	Artikelnummer eingeben	Artikelnummer	
2	Preis anzeigen		Preisanzeige
3	Kauf stornieren	Stornierung	
4	Betrag auswerfen		Betrag ausgeworfen
5	Betrag entnehmen		

Code

V-Modell: Testen auf verschiedenen Ebenen



Vollständiges Testen: Wie viele Testfälle sind nötig?



- Anweisungs-Abdeckung
- Zweig-Abdeckung
- Pfad-Abdeckung


Testfälle (System-Testfall = Black Box)

■ Abstrakter Testfall:

Testfall 1	Name: Stornieren		
	Vorbedingung: keine		
Nr.	Testschritt	Eingabedaten	Soll-Ergebnis
1	Artikelnummer eingeben	gültige Artikelnummer	
2	Preis anzeigen		Preisanzeige
3	Kauf stornieren	Stornierung	
4	Betrag auswerfen		Betrag ausgeworfen
5	Betrag entnehmen		

■ Konkreter Testfall:

Testfall 1.a	Name: Stornieren		
	Vorbedingung: keine		
Nr.	Testschritt	Eingabedaten	Soll-Ergebnis
1	Artikelnummer eingeben	Artikelnummer = 123	
2	Preis anzeigen		Preisanzeige: 1,00€
3	Kauf stornieren	Stornierung	
4	Betrag auswerfen		Betrag = 0 ausgeworfen
5	Betrag entnehmen		

- 
- Ansätze für händisches modellbasiertes Testen
 - Warum eine Automatisierung praktisch schwierig ist
 - Ergebnisse eines Experiments: Fehler beim Testfall-Entwurf

- **Automatisierte Testfall-Herleitung**
- Automatisierte Testfall-Ausführung

Definition: Testbarkeit

Testbarkeit (eines UML-Modells): Es sind alle Informationen enthalten, die für die Ableitung des Testfalls nötig sind.

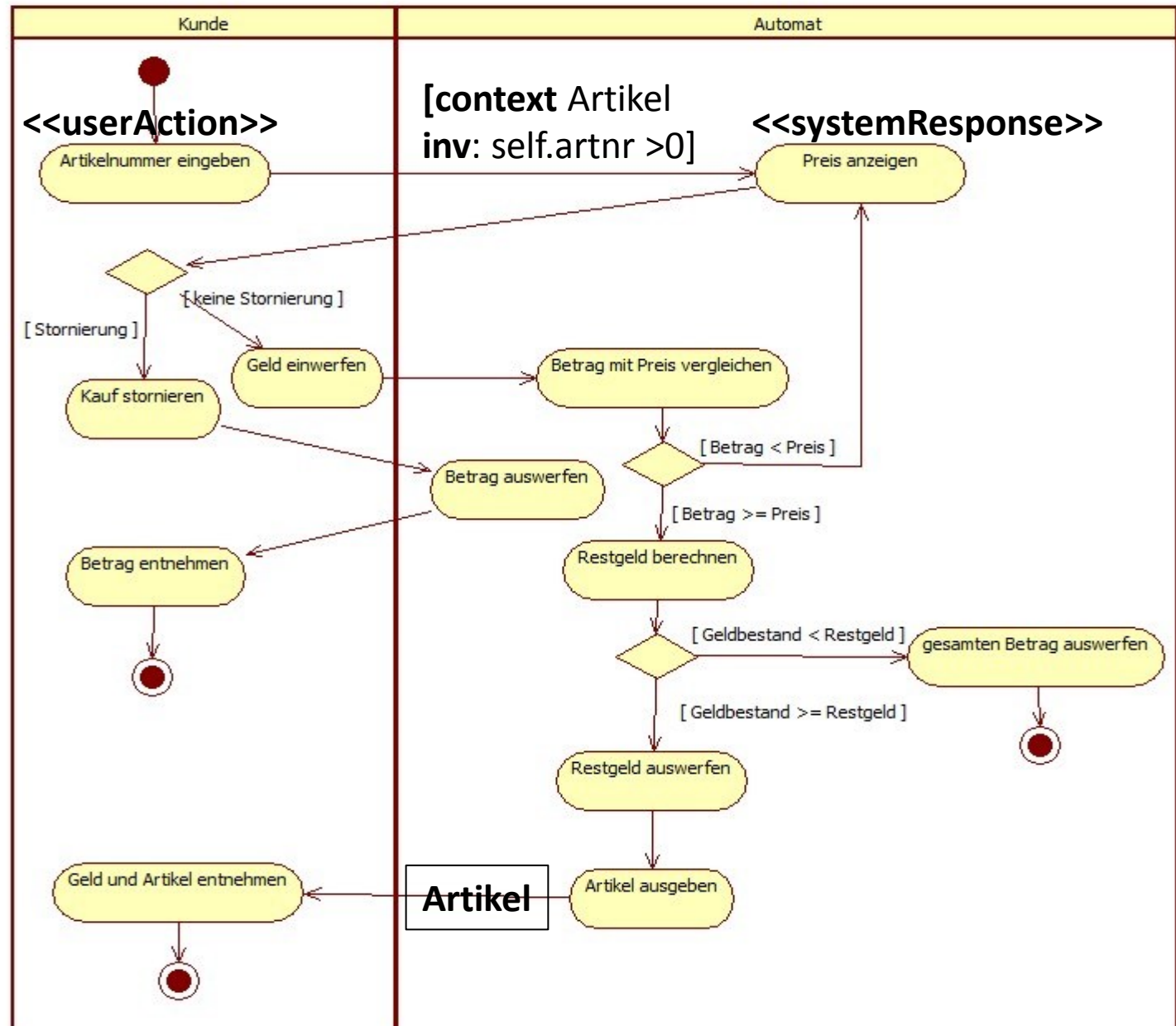
Testfall 1	Name: Stornieren		
	Vorbedingung: keine		
Nr.	Testschritt	Eingabedaten	Soll-Ergebnis
1	Artikelnummer eingeben	gültige Artikelnummer	
2	Preis anzeigen		Preisanzeige
3	Kauf stornieren	Stornierung	
4	Betrag auswerfen		Betrag ausgeworfen
5	Betrag entnehmen		

- **Inhalte:**
 - Vorbedingung, Testschritte, Eingabedaten und erwartetes Ergebnis
- **Detailltiefe:**
 - dieselbe Detailtiefe von Modell und Testfälle, z.B. Aktivität = Testschritt
- **Kontrollfluss:**
 - u.a. alle Fehler- und Sonderfälle

Aktivitätsdiagramm nicht vollständig testbar!

Vervollständigung durch...

- OCL-Ausdrücke für Bedingungen
- Stereotypen
- Objektfluss



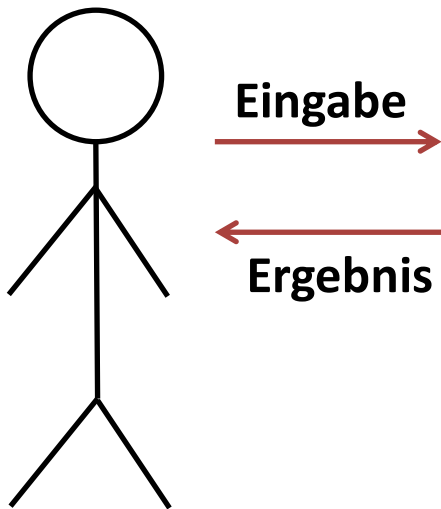
Praktische Probleme

- Testbarkeit der Anforderungen war nicht Ziel, sondern Verständlichkeit -> Details weggelassen
- Tester / Spezifizierer sind oft Key User, also keine Testexperten
- Sonder- und Fehlerfälle nicht unbedingt vollständig
- Vollständigkeit
 - Tests finden 30-60% der Fehler
 - 15% der Fehlern ausgeliefert

Automatisierung beim Testen

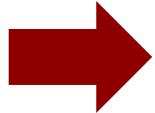
- Automatisierte Testfall-Herleitung
- **Automatisierte Testfall-Ausführung**

Automatisierte Testfall-Ausführung?



Überblick des Vortrags

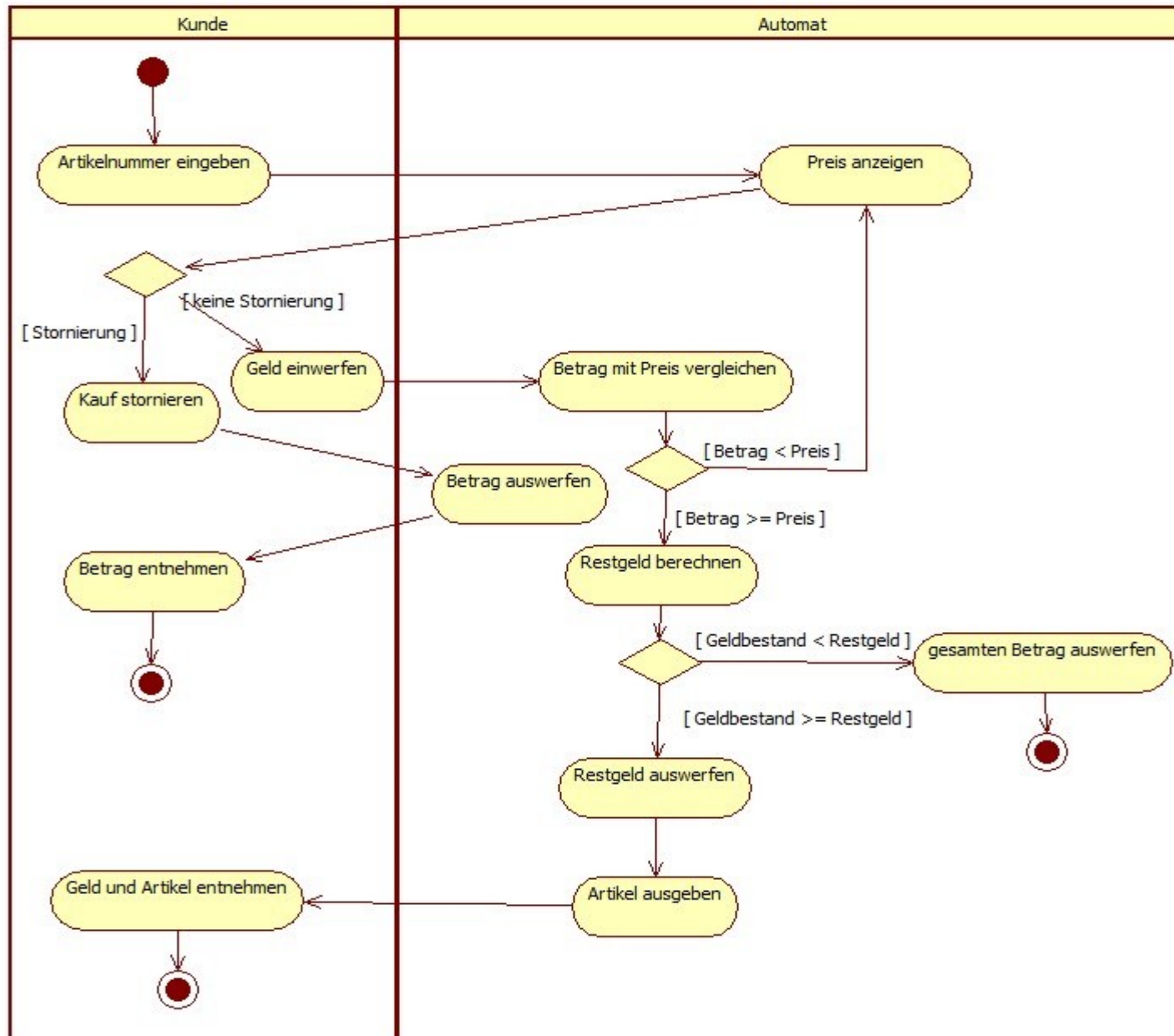
- Ansätze für händisches modellbasiertes Testen
- Warum eine Automatisierung praktisch schwierig ist
- Ergebnisse eines Experiments: Fehler beim Testfall-Entwurf



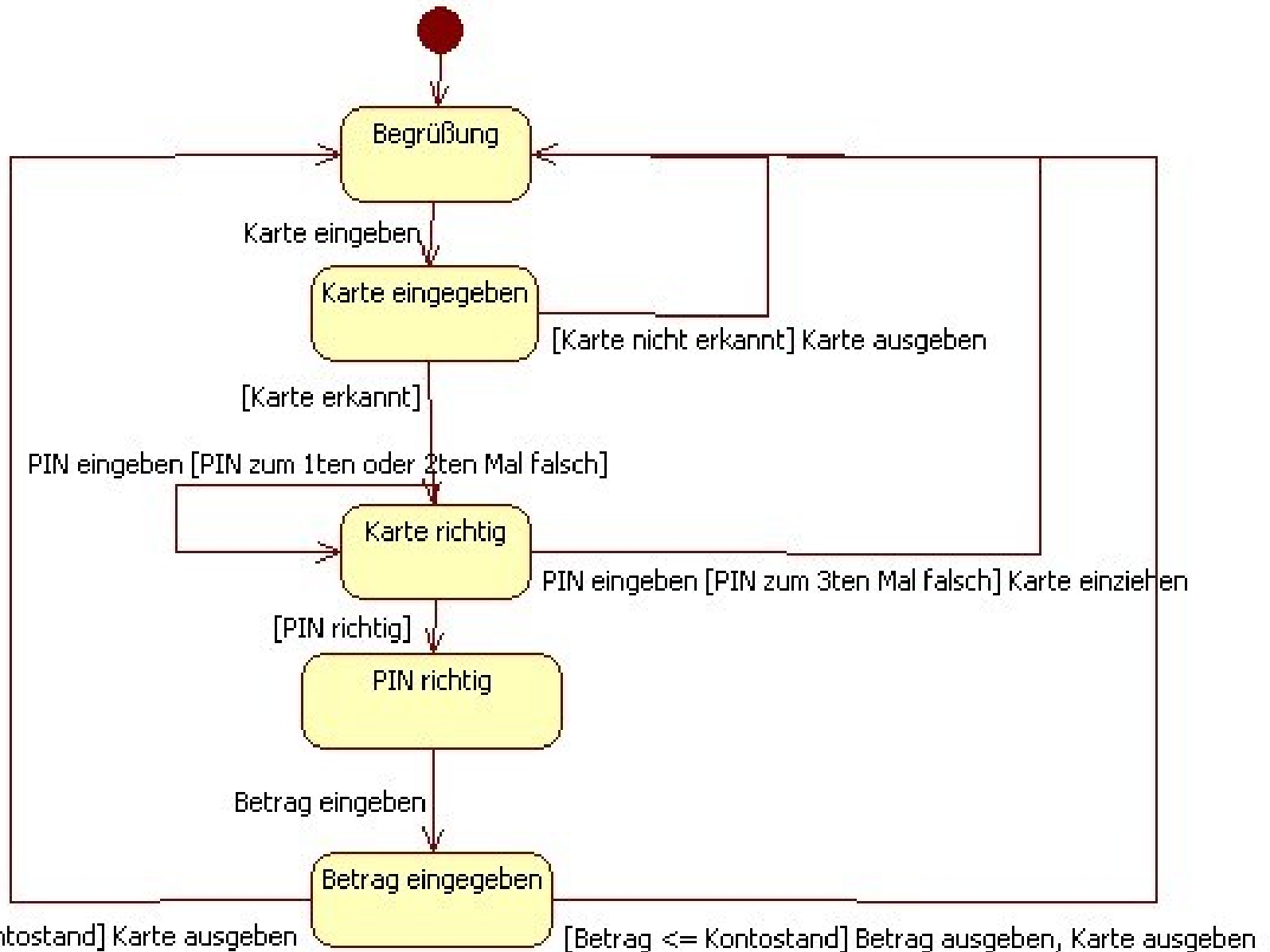
Studenten-Experiment

- 84 Teilnehmer/innen in 3 Gruppen an 2 Hochschulen (A. Herrmann, M. Felderer)
- Vorkenntnisse:
 - Anwenderwissen (Getränkeautomat und Geldautomat)
 - UML aus vorigem Kurs
 - Testen: Einführung, Übungsbeispiel mit Musterlösung
- Testbarkeit der UML-Modelle:
 - Detailtiefe: wie Systemtests
 - Kontrollfluss: vollständig, alle Sonderfälle
 - Inhalte: Vorbedingungen, Eingabedaten und erwartete Ergebnisse nur teilweise im Modell

Aktivitätsdiagramm Getränkeautomat



Zustandsdiagramm Geldautomat

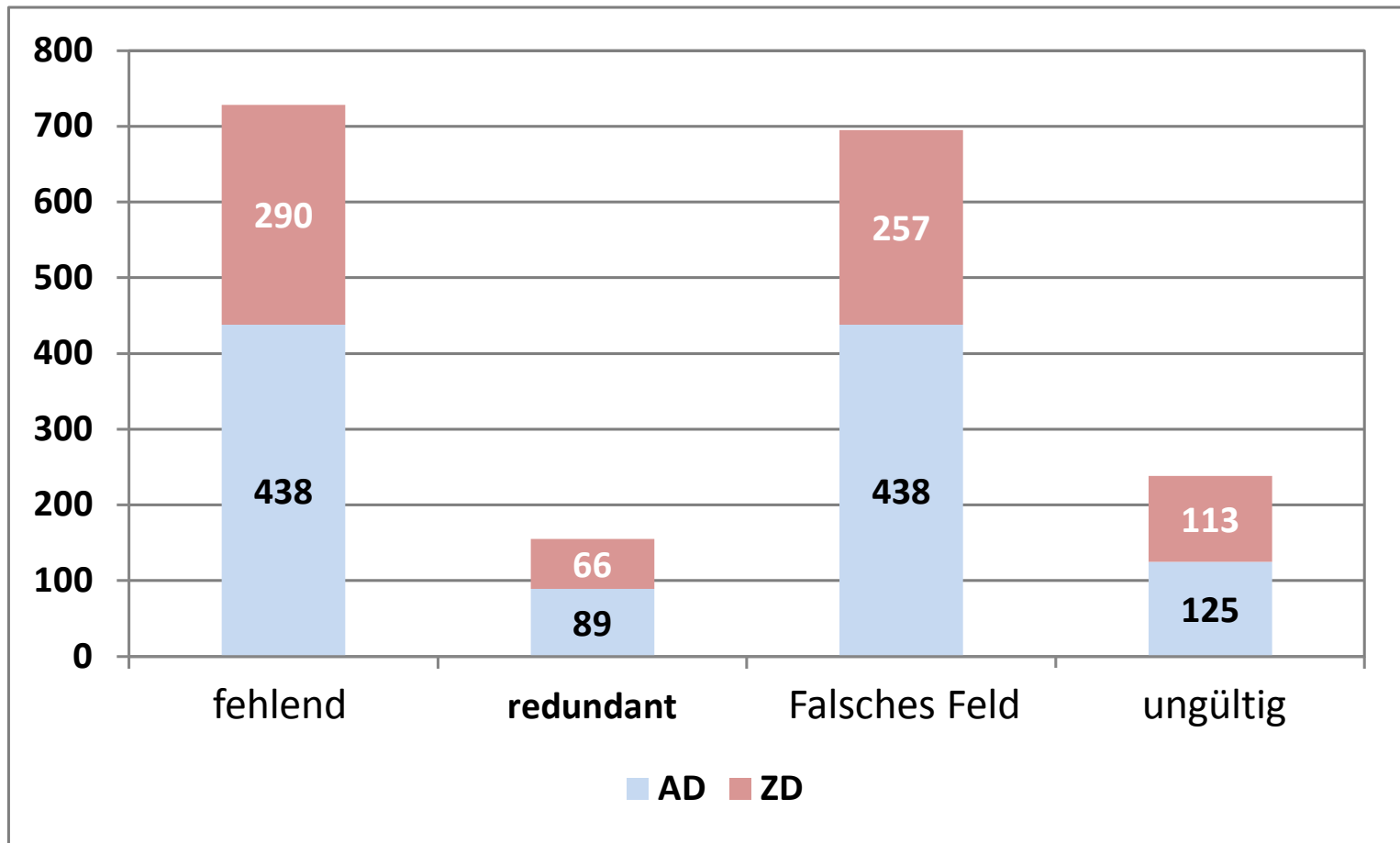


Gruppen im Experiment

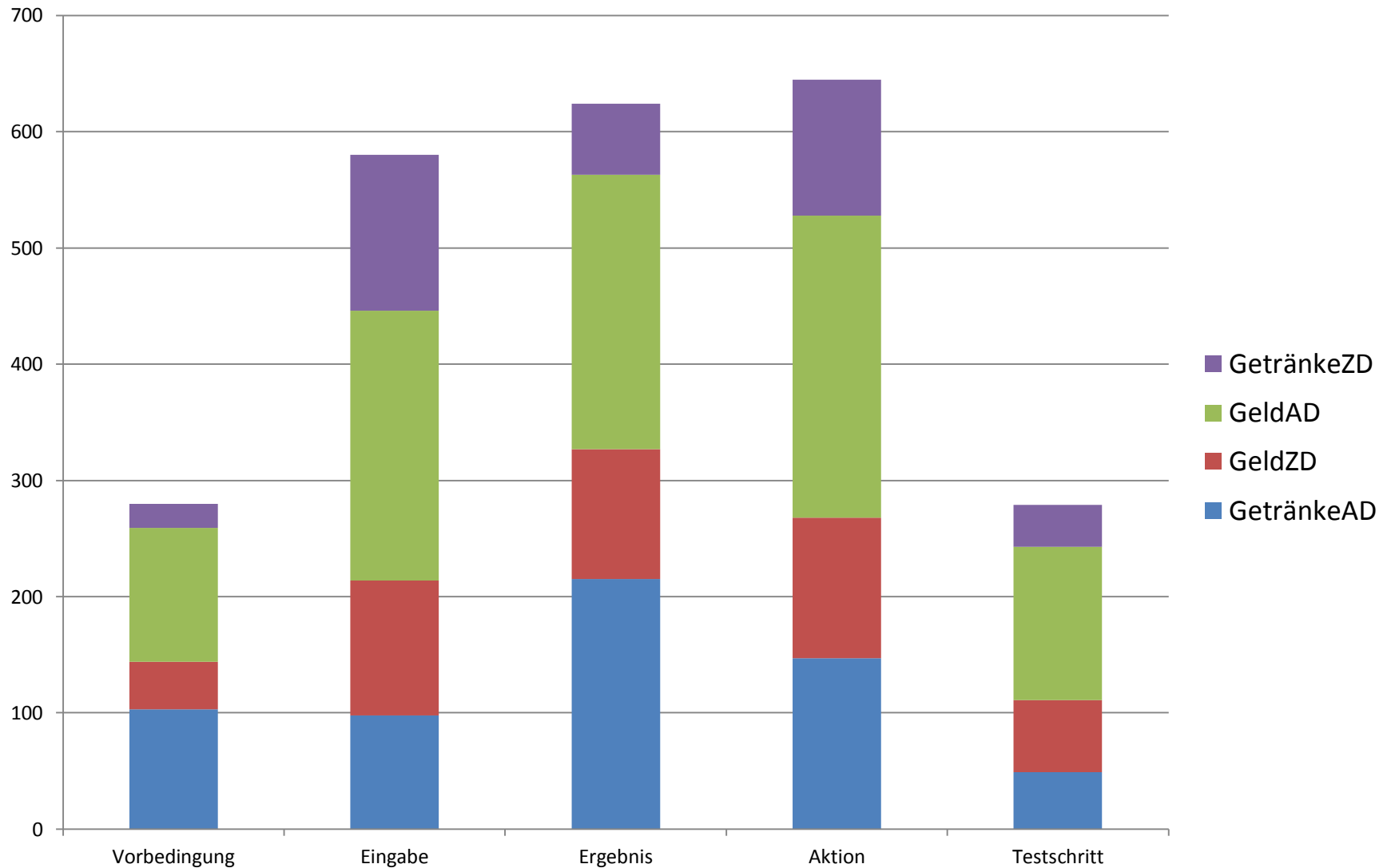
	Getränke-Automat	Geld-Automat
Aktivitäts-Diagramm	Gruppe A	Gruppe B
Zustands-Diagramm	Gruppe B	Gruppe A

Ergebnisse: Welche Fehler?

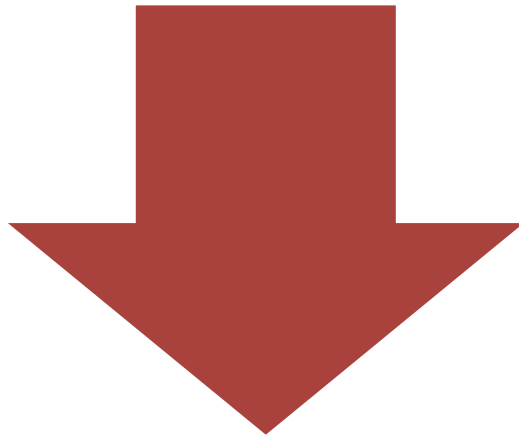
- 150 Sätze von Testfällen, ca. 340 Testfälle
- 1816 Fehler:



Ergebnisse: fehleranfällige Felder?



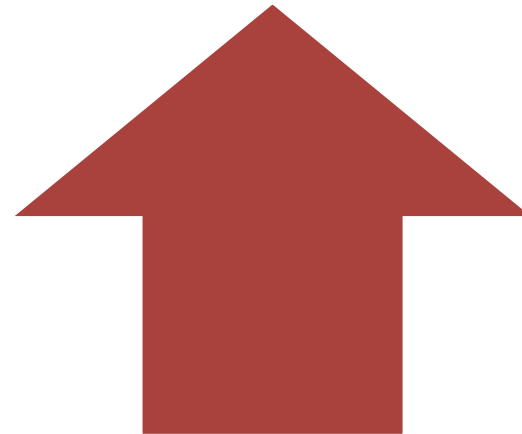
Ergebnisse: Vergleich der Diagramme



AD als verständlicher
empfunden & besser
verstanden (=richtigere
Antworten auf
Verständnisfragen)



Aktivitäts-
diagramm: mehr
Fehler



Schlussfolgerungen

- UML-Modelle müssen vollständig sein, auf „intuitive“ Ergänzung sollte man sich nicht verlassen
- Vermutlich hilfreich: konkrete Regeln wie „Eine Aktivität entspricht einem Testschritt.“
- Wahl des Diagramms:
 - Aktivitätsdiagramm besser geeignet für Kommunikation über Anforderung (mit Kunden)
 - Zustandsdiagramm für Testfall-Ableitung

Heute Nachmittag 14:00-18:00 Uhr

- Testfälle von Hand erstellen für...
 - Zustandsdiagramm
 - Aktivitätsdiagramm
 - textueller Use Case
- Diskussion des Vorgehens

Vielen Dank für Ihre Aufmerksamkeit!

Fragen?



Quellen der Bilder

- http://commons.wikimedia.org/wiki/File:Soft_drink_vending_machine_in_Japan_01.jpg