A silhouette of a knight on a horse, holding a flag, set against a dramatic, cloudy sky. The knight is positioned on the left side of the frame, facing right. The horse is rearing up slightly. The sky is filled with soft, white clouds, and a bright light source is visible behind the knight, creating a silhouette effect.

Auf dem **Kreuzzug** gegen  
**Fehler** (in Software)

by Jerome Müller

A stage with red curtains and a wooden floor. The curtains are pulled back, revealing a dark stage floor. The text is centered on the stage.

# Ein Stück in drei Akten

I - Der “Fehler-Entwicklungs-Prozess”

II - Agilität und Testing

III - Von der Idee zur Realität

Wo kommen die  
Fehler her?



# Teil I:

## Der “Fehler-Entwicklungs-Prozess”

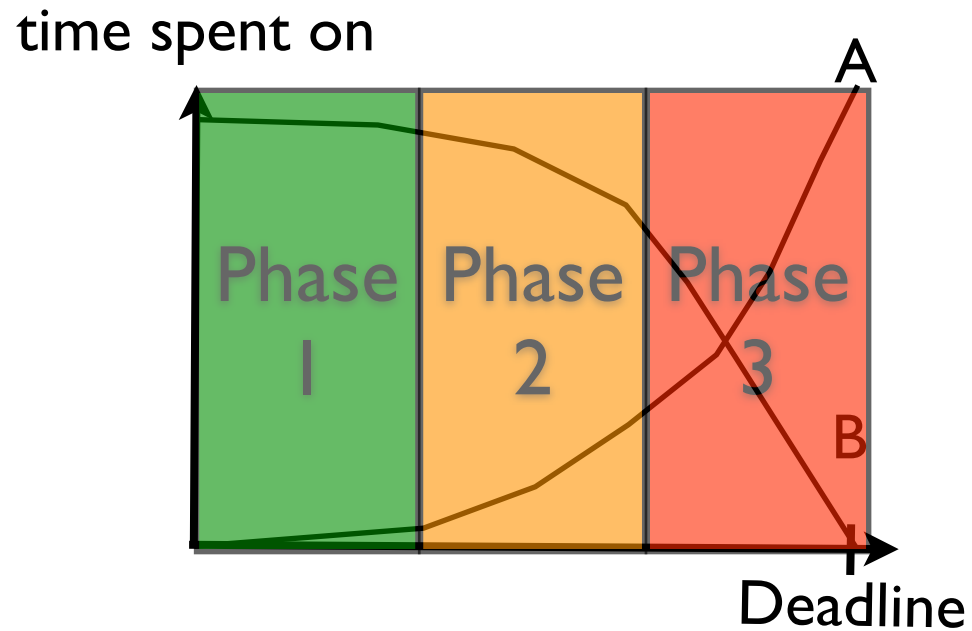
Dinge die scheinbar funktionieren -  
aber Fehler garantieren

Problem 1:

Bugzilla! Jira!

Als Kommunikationstool zwischen  
Entwicklern und Testern

# Problem 2:

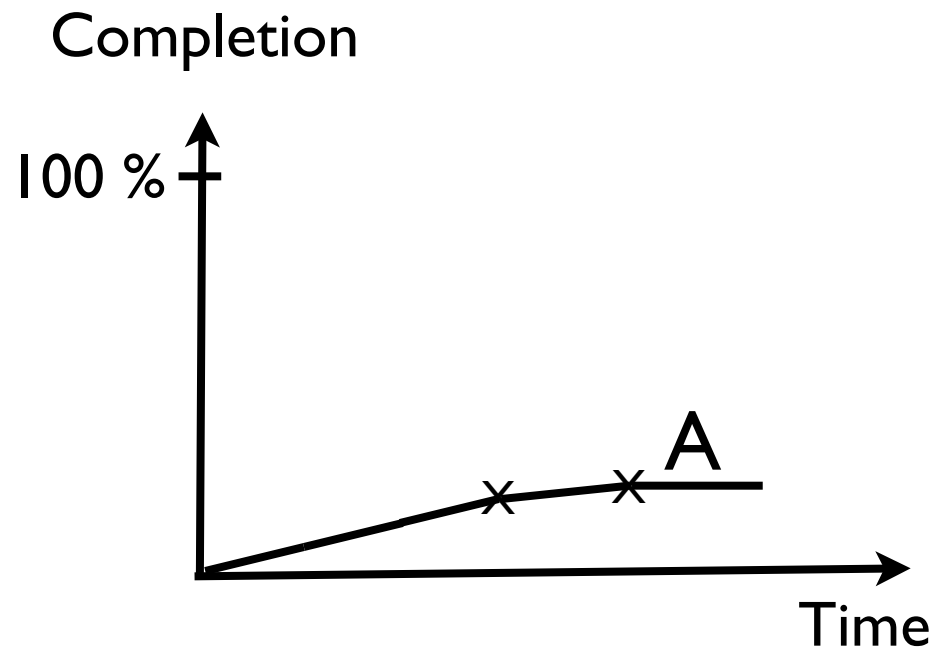


A = Coding

B = Anforderungen verstehen

# Problem 3:

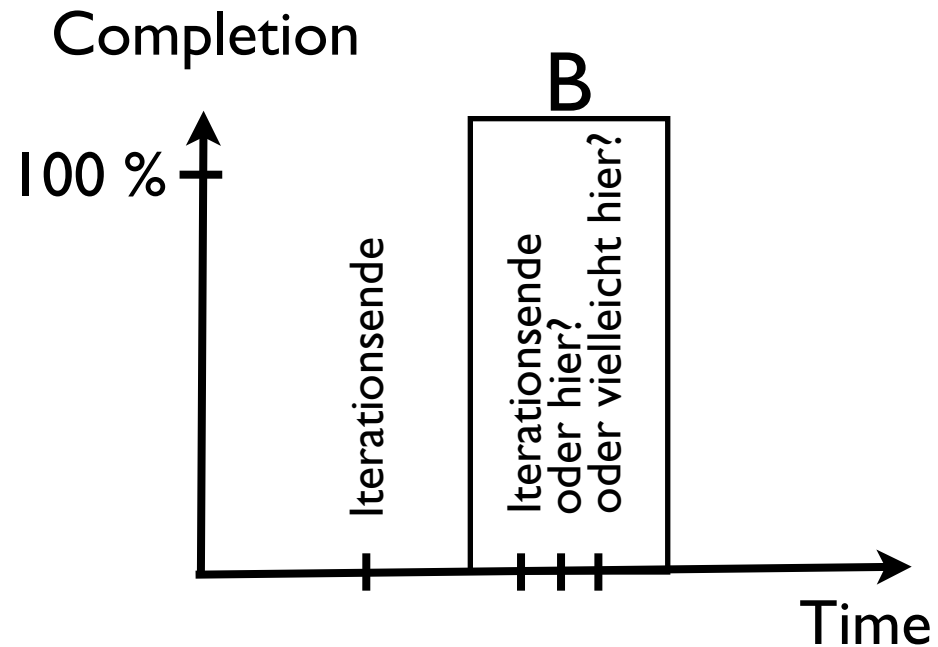
Bugfix “Iterationen”



A = Bugfix Iteration

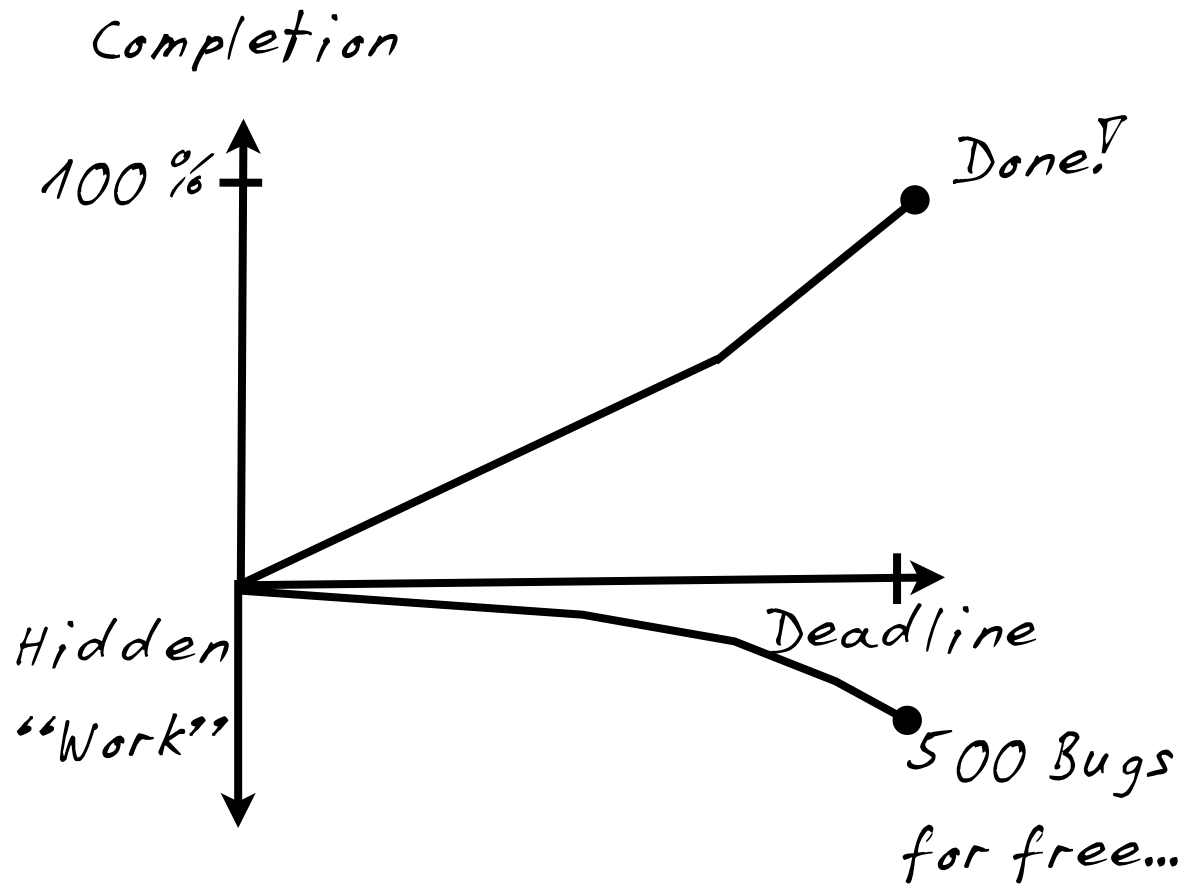
# Problem 4:

“Flexible” Iterationen



**B = Flexible Iteration**

# Problem 5: Fertig?



# Problem 6: <sup>Datenbank</sup> Konfigurierbar

Zu testende Software ist für alles designed!

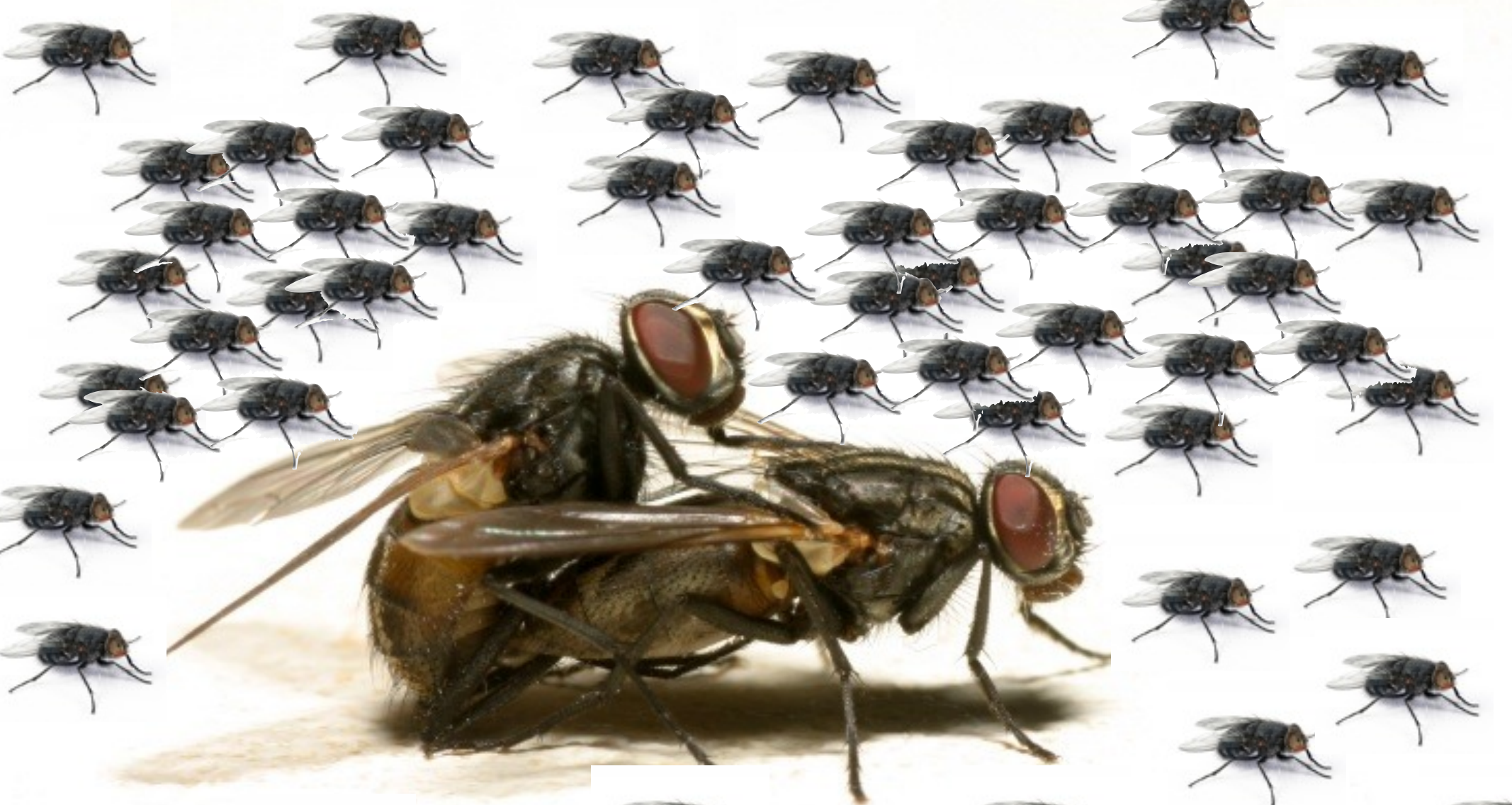
*Extensibility*  
*Services*  
*Maintainability*

*Scalability*  
*Plattformunabhängigkeit*

*Reliability*  
*Starchheit*  
*Compliance*  
*Agility*

...aber nicht um getestet zu werden...

Diese Verhalten  
sind...



...Brutstätten für Bugs

Wie können wir diese  
Probleme  
verhindern?





Teil II:

Agilität und Testing

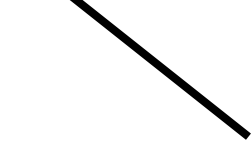
Auf der Jagd nach Bugs

# Agiles Einmal Eins: Iterationen

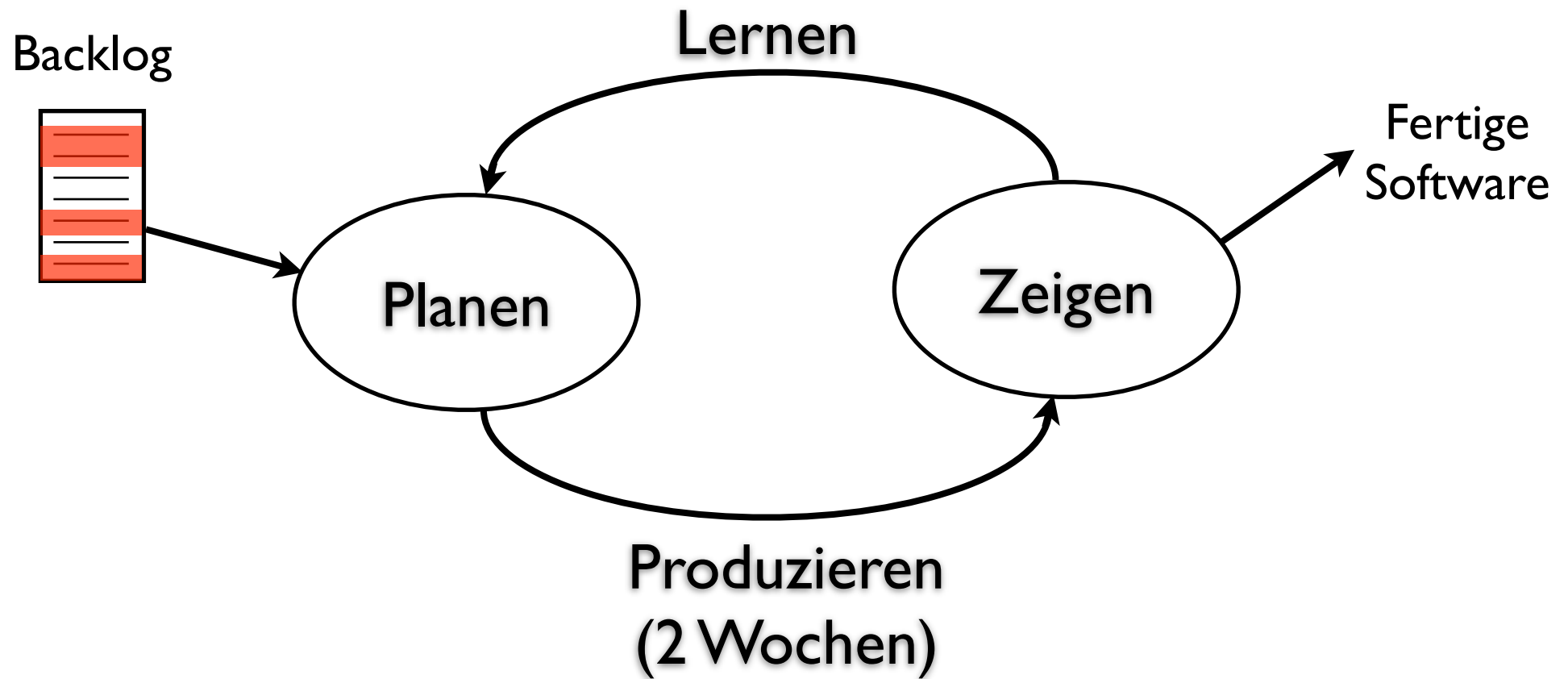
1 - Timeboxed

2 - Endet mit FERTIGER Software

*Impliziert dass die Software getestet wurde!*



# Scrum



Aber wann kommt  
die Testphase?

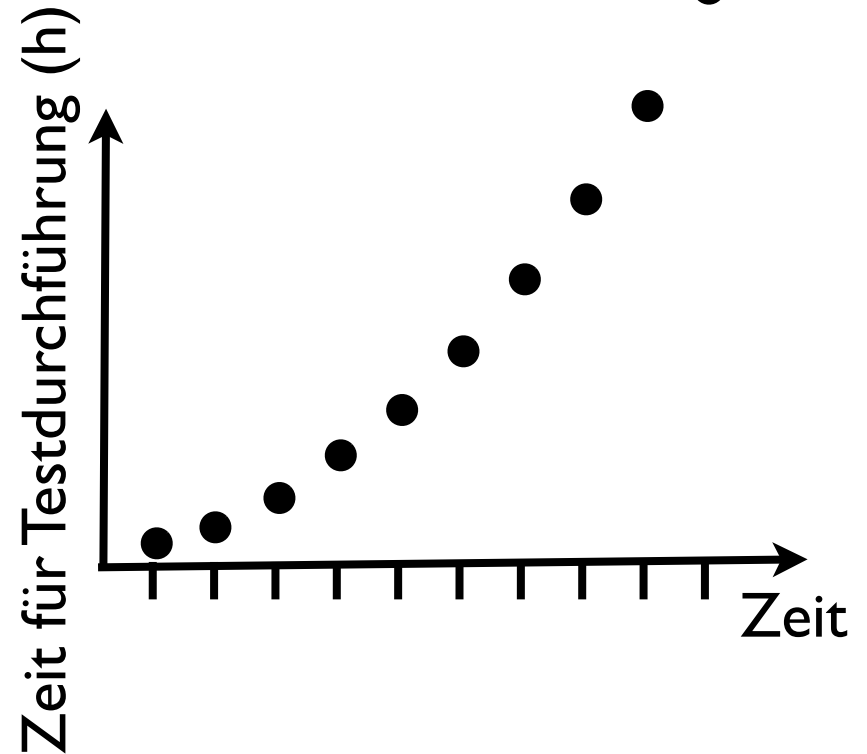
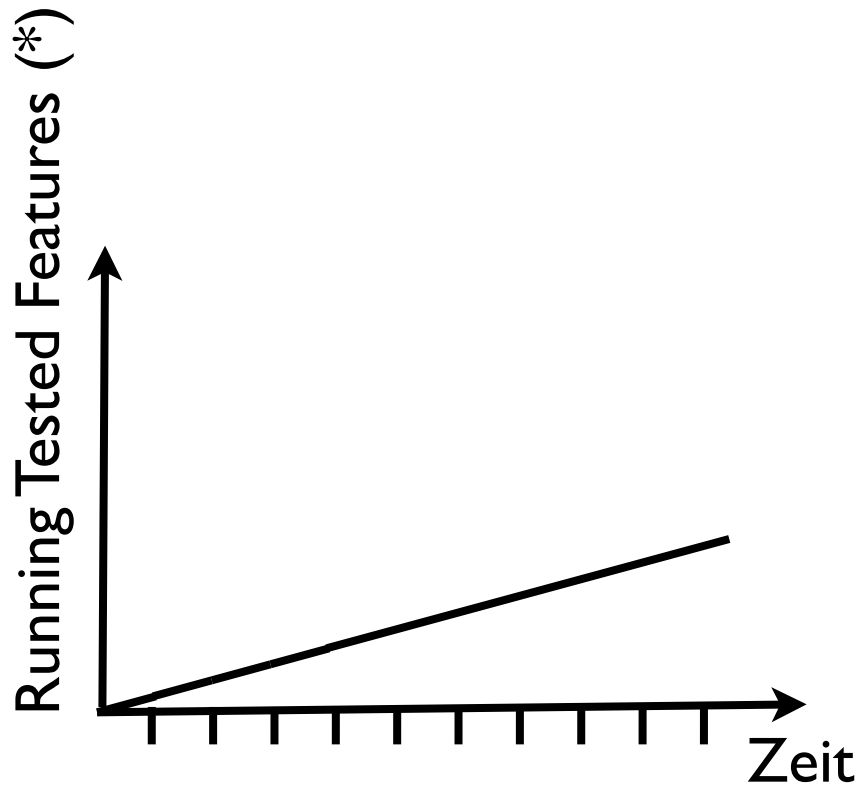
# Scrum:

Kurzbeschreibung von Ron Jeffries

1. Produce Done-Done software on a regular basis (\*)
2. Remove every obstacle to doing item 1.

(\*) 2 - 4 Wochen.

Und wann wird getestet?



Automatisierte Tests  
müssen Bestandteil der  
Entwicklungsmethode  
werden!!!

# Requirements.doc

Wozu?

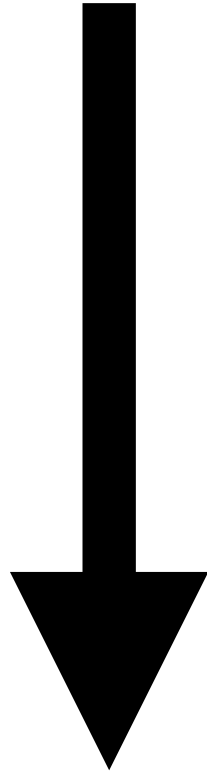
Planen + Lehrmittel + Beweisen

# Requirements the Agile Way

Planen = Feature Beschreibung

Lehren = Konversation

Beweisen = Automatisierte Tests



Feature

Konversation

Test(s)

Code

# “Fertig”

1 Feature => X Tests => Alle Grün

Das ist nur  
der **erste Schritt.**

Wie gehts **weiter?**





Teil III:  
Von der Idee zur Realität

# Idee zur Realität



x

In welcher Zeiteinheit  
wird x in ihrem Projekt gemessen?

“Demo! Heute!”

Was funktioniert?

Wie lang bis Demo?

Bug Risiko?

Unit Tests  
Check In Regeln

Automatische Tests  
Kleine Features

Automatisches  
Deployment

Code

Testing

Deploy

---

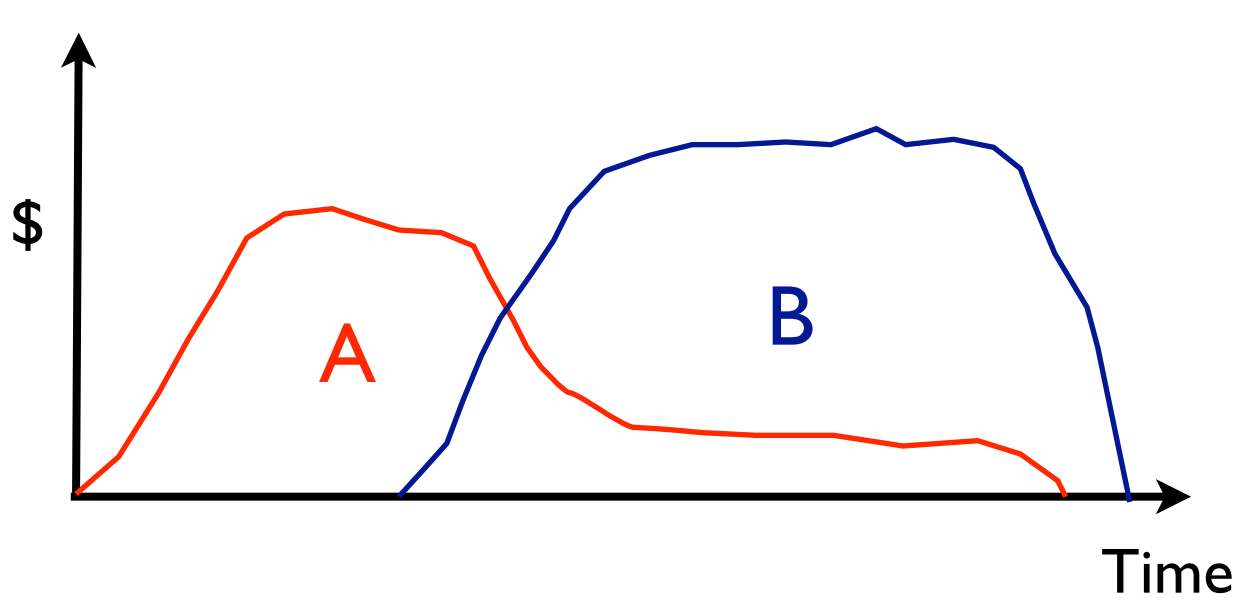
Integration

Daten Migration

Automatischer Build  
Continuous Integration

Diff Scripts  
Rollback Scripts

Warum wir  
bezahlt werden



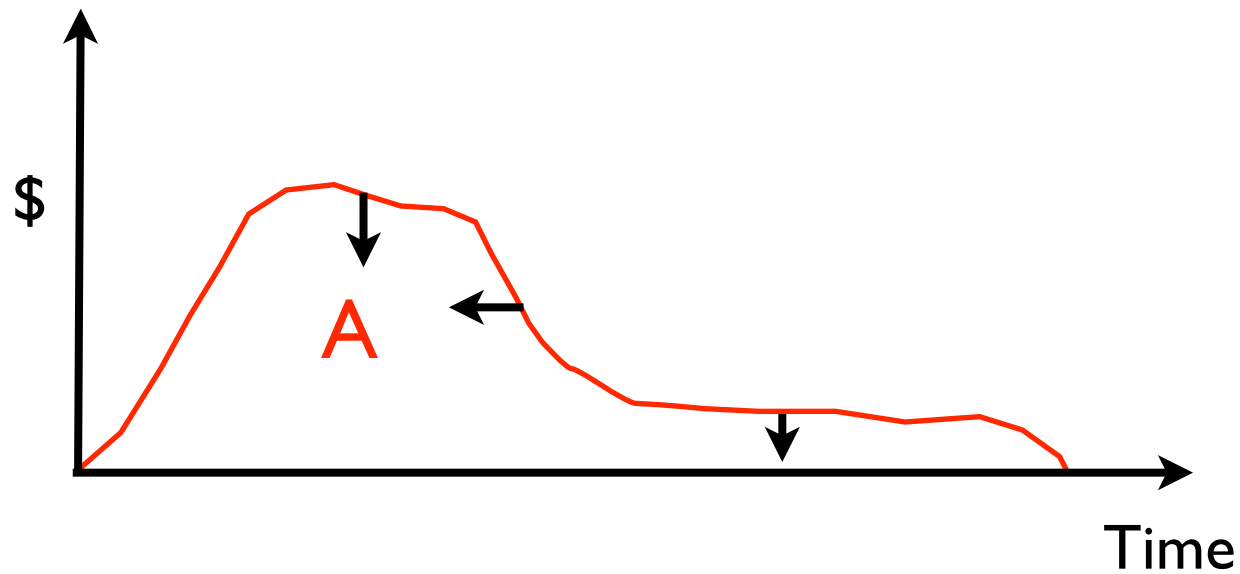
A = Kosten

B = Nutzen

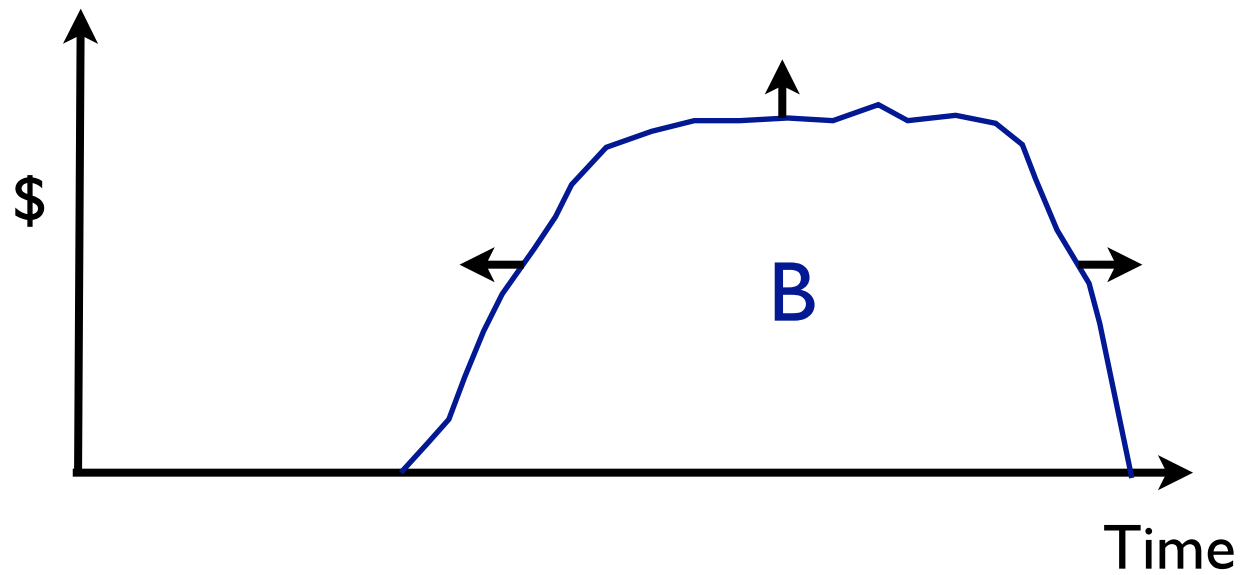
$B > A$

# Billiger machen

A = Kosten



# Mehr Nutzen



B = Nutzen

# Testphase?

Code

Testing

Deploy

---

Integration

Daten Migration

blau = Aufgabe Entwicklungsteam

rot = Aufgabe vom Testteam

Angenommen es ist  
wahr, dass...

...der Prozess falsch ist.

...wir Bugs verhindern können.

...es keine Testphase braucht.

Wie verändert  
sich dadurch der Job der  
Tester?

