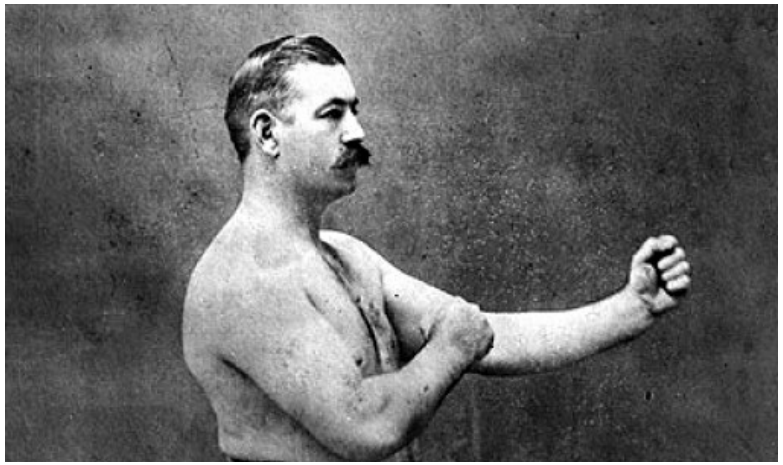


Softwareprozesse: Timeboxing & Continuous Delivery

BENJAMIN PROJDAKOV

The Timeboxing Process Model

- Voller Titel: The Timeboxing Process Model for Iterative Software Development
- Verfasser:
  - Professor Pankaj Jalote

    Department of Computer Science and Engineering

    Indian Institute of Technology

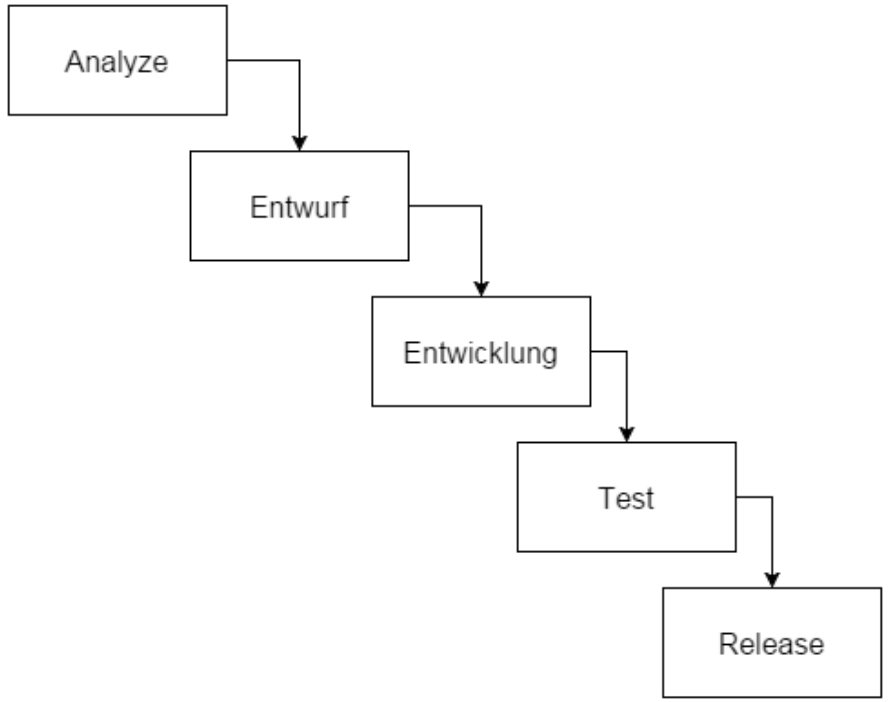
    Kanpur
  - Aveejeet Palit, Priya Kurien

    Infosys Technologies Limited

    Bangalore

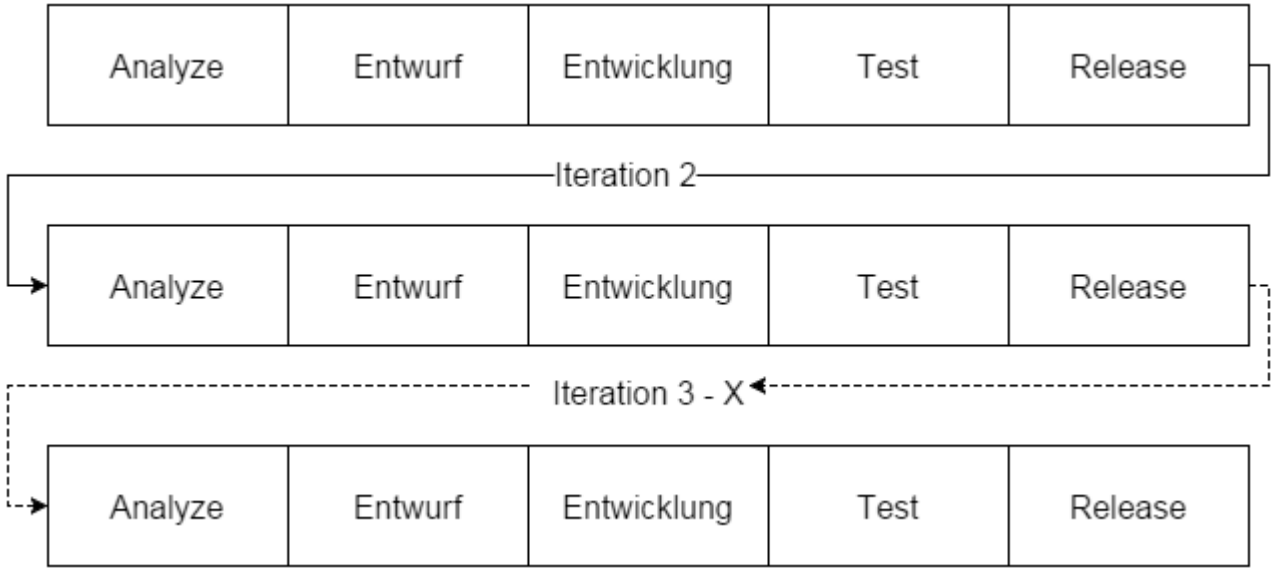
Einführung: Wasserfallmodell

- Linear
- Nicht iterativ
- Lange Entwicklungsdauer typisch
- Anforderungen:
  - Müssen vollständig bekannt sein
  - Änderungen schwierig
- Entwicklung in Phasen
- Ergebnis: Vollständige Software
- Komplexität der Software eher gering



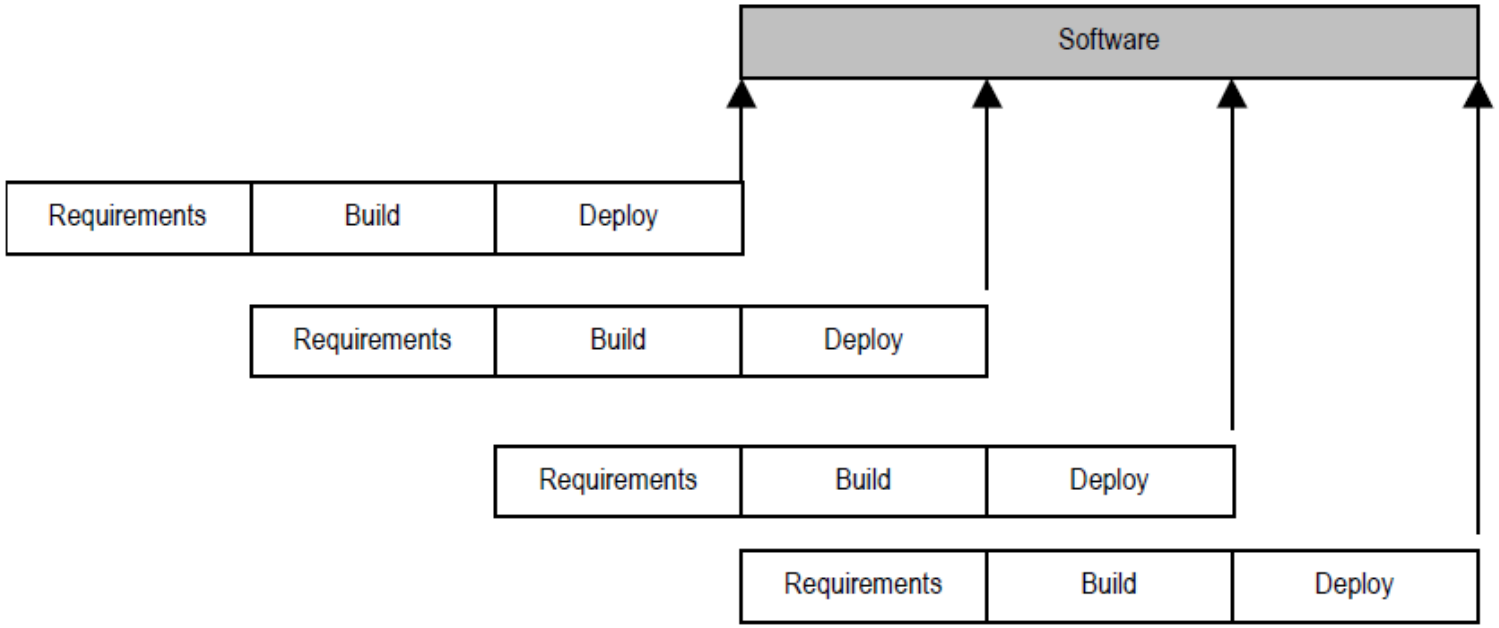
Einführung: Iterative Modelle

- Linear
- Entwicklung:
  - In Zyklen
  - Zyklus unterteilt sich in Phasen
- Kurze Entwicklungsdauer pro Zyklus
- Anforderungen:
  - Müssen pro Iteration bekannt sein
  - Änderungen in nächster Iteration möglich
- Ergebnis: Funktionierende Software nach jedem Zyklus



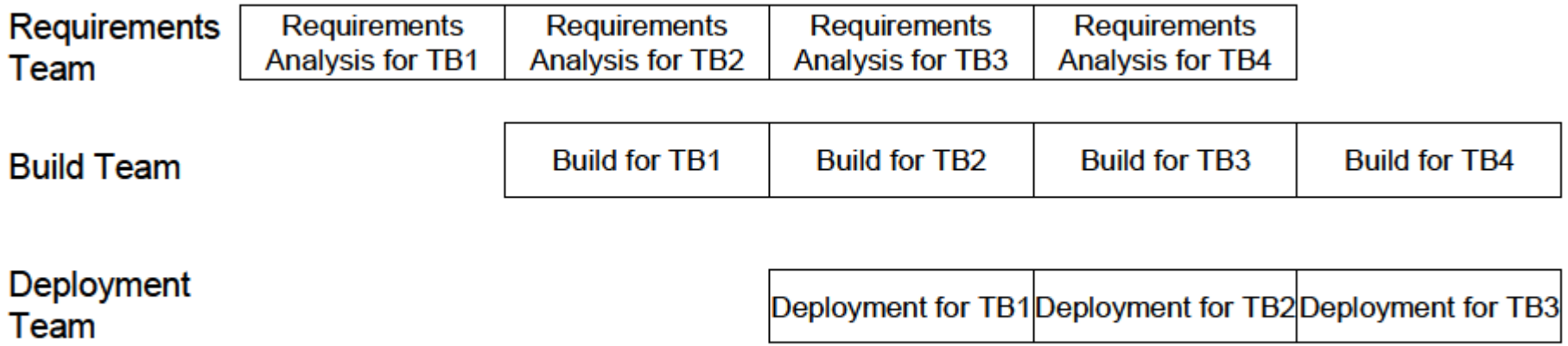
Timeboxing Modell

- Entwicklung:
  - In Zyklen
  - Zyklus unterteilt sich in Phasen
  - Ausführung mit Versatz parallel
- Feste Entwicklungsdauer pro Zyklus
  - Wochen bis Monate
- Anforderungen:
  - Müssen pro Iteration bekannt sein
  - Änderungen in nächster Iteration
- Ergebnis:
  - Häufige Releases
  - Funktionierende Software nach jedem Zyklus



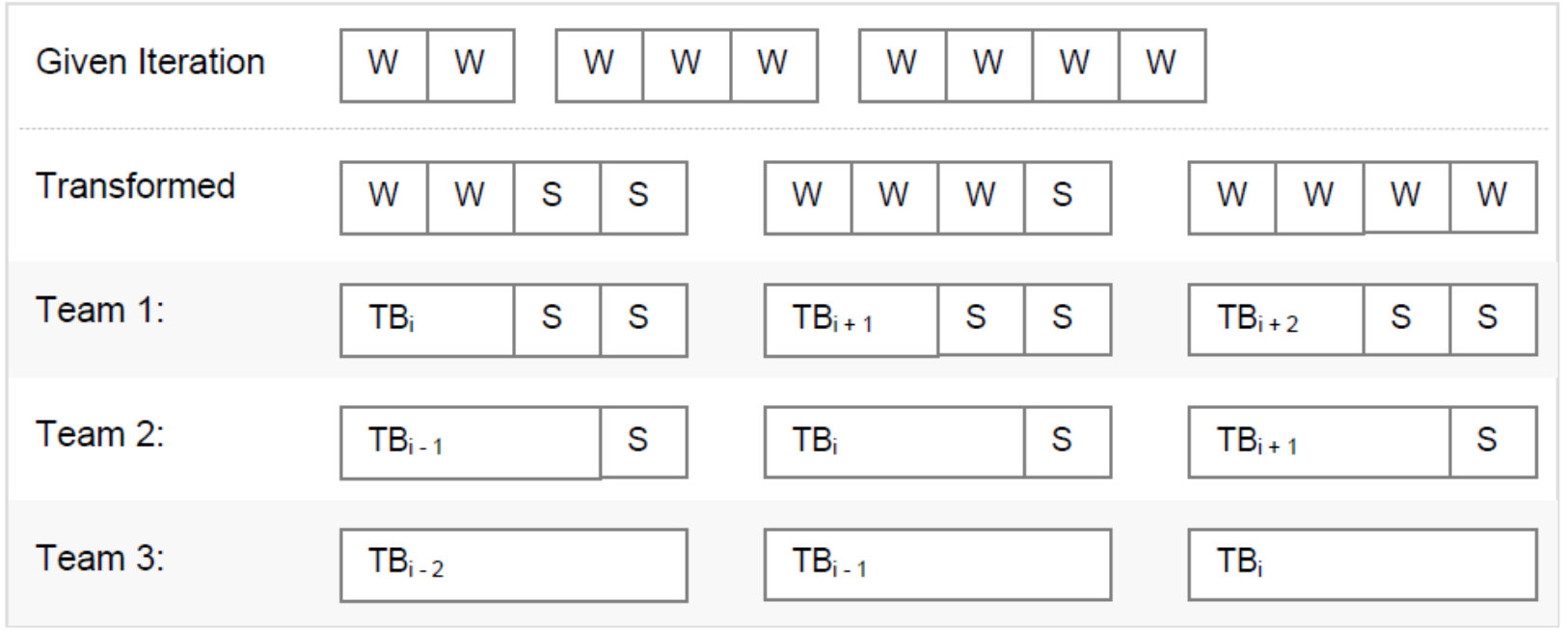
Timeboxing Modell: Phasen & Teams

- Phasen:
  - Anzahl Variabel
  - Gleiche Dauer ideal
- Teams:
  - Eine Aufgabe pro Team
  - Größe abgestimmt auf Dauer



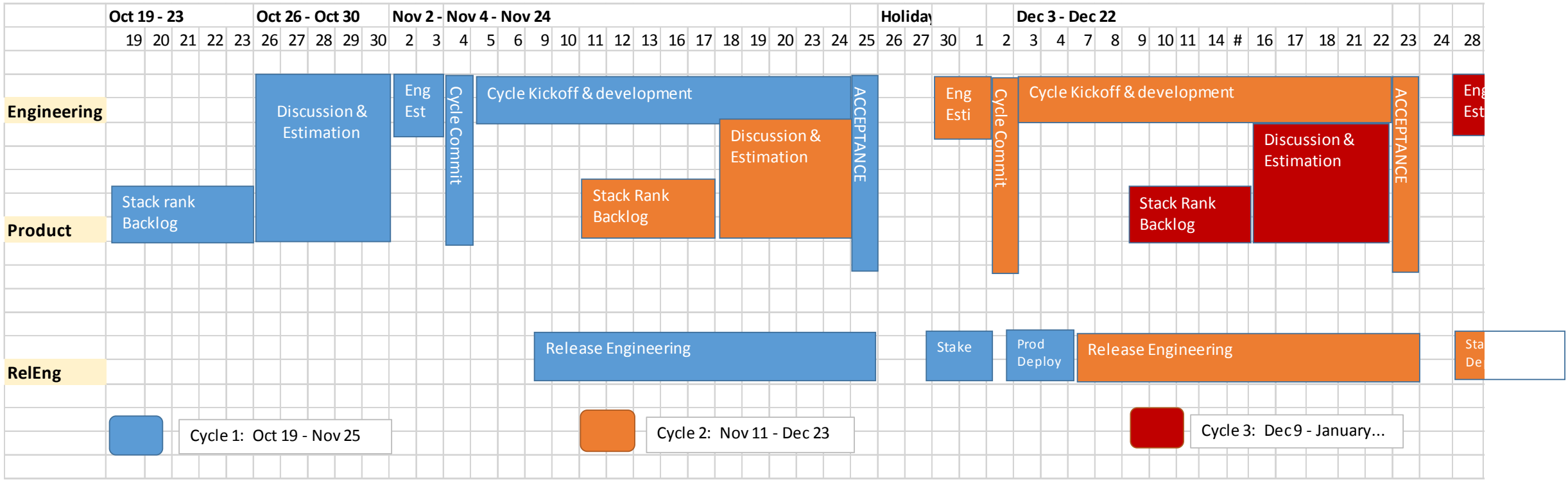
Timeboxing Modell: Phasen & Teams

- Ungleiche Boxen führen zu ineffizienter Teamauslastung
- Lösungen:
  - “Right-scaling” von Teams
  - Ressourcenumverteilung
- Kann auch durch Verzögerungen in einer Box verursacht werden

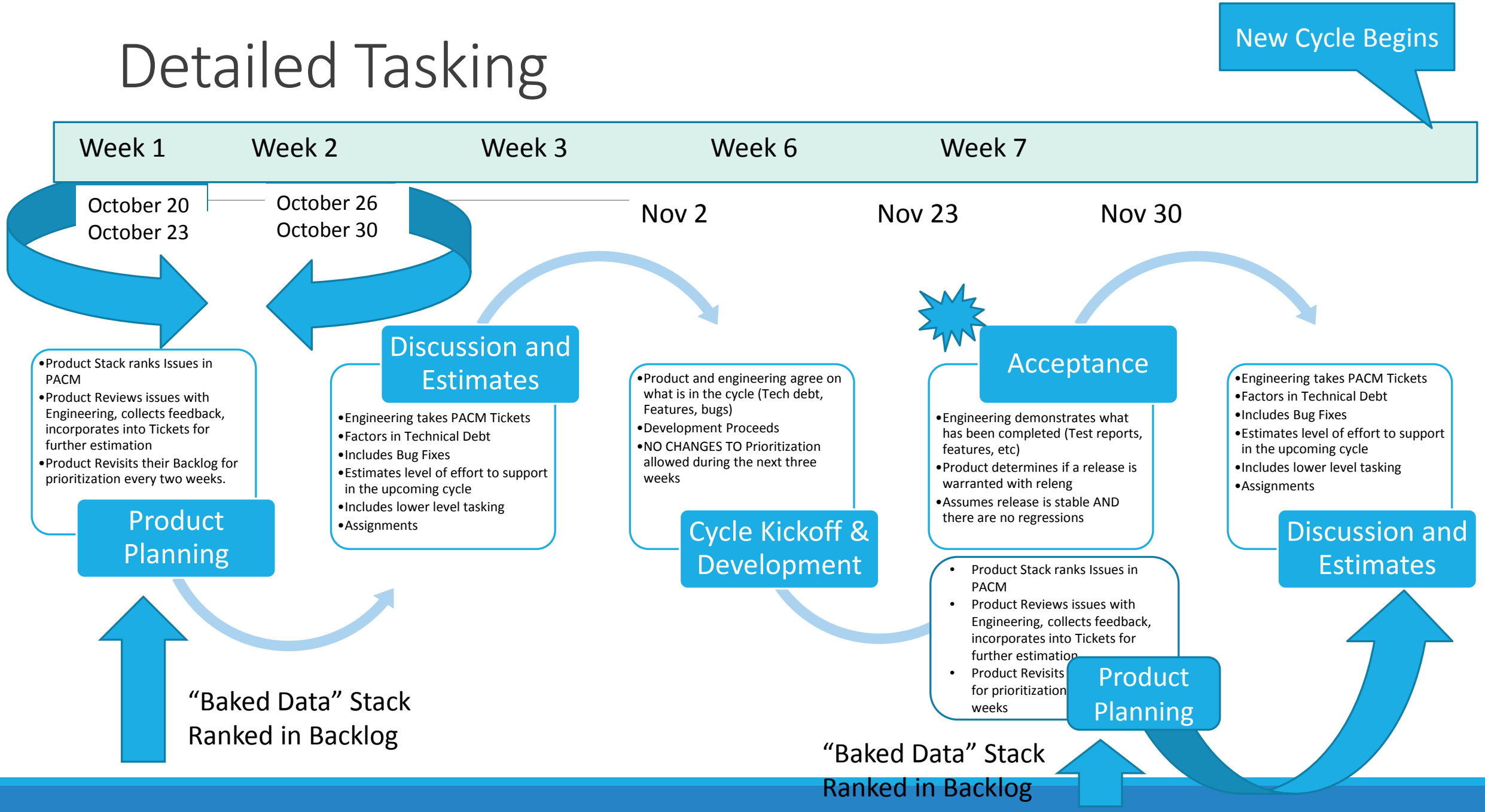


W represents “work” and S represents “slack”

Timeboxing Modell: Reales Beispiel



Detailed Tasking

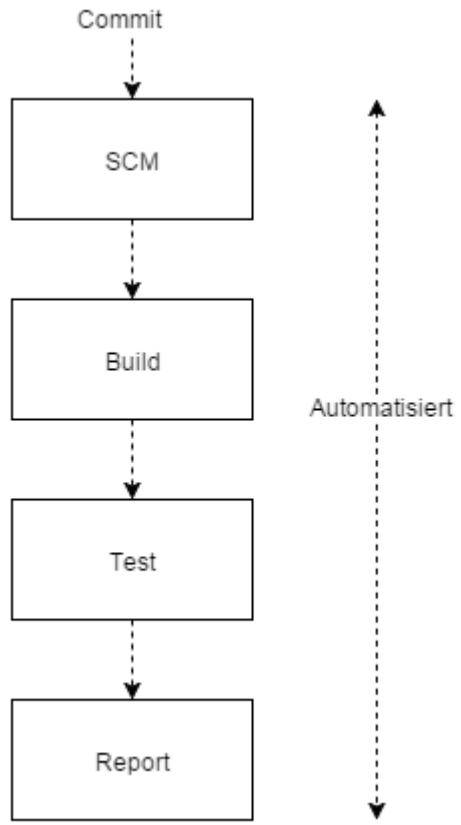


Continuous Delivery

- Voller Titel: Release Management with Continuous Delivery: A Case Study
- Verfasser:
  - A. Maruf Aytekin, Release Management Team atCentral Securities
  - Depository Institution (MKK),
  - Istanbul, TurkeyDepartment of Computer Science and Engineering
- Veröffentlicht:
  - World Academy of Science, Engineering and Technology
  - International Journal of Social, Education, Economics and Management Engineering Vol:8, No:9, 2014

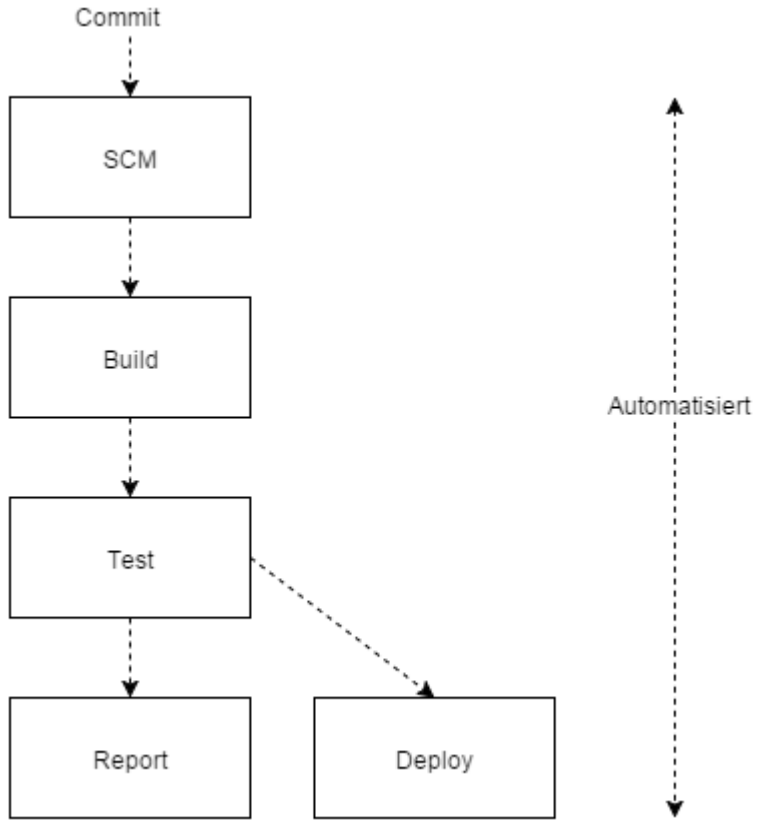
Einführung: Continuous Integration

- Entwicklungsprozess
- Voraussetzungen:
  - Häufige Integration
  - Automatisierung von “Builds”, Testläufen, Reporting, QA, …
  - Kurze Testzyklen
  - Schnelle Laufzeiten
  - Gespiegelte Produktionsumgebung für Testing
- Vorteile
  - Schnelles Feedback
  - Weniger Aufwand für Defektkorrekturen
  - Höhere Software Qualität



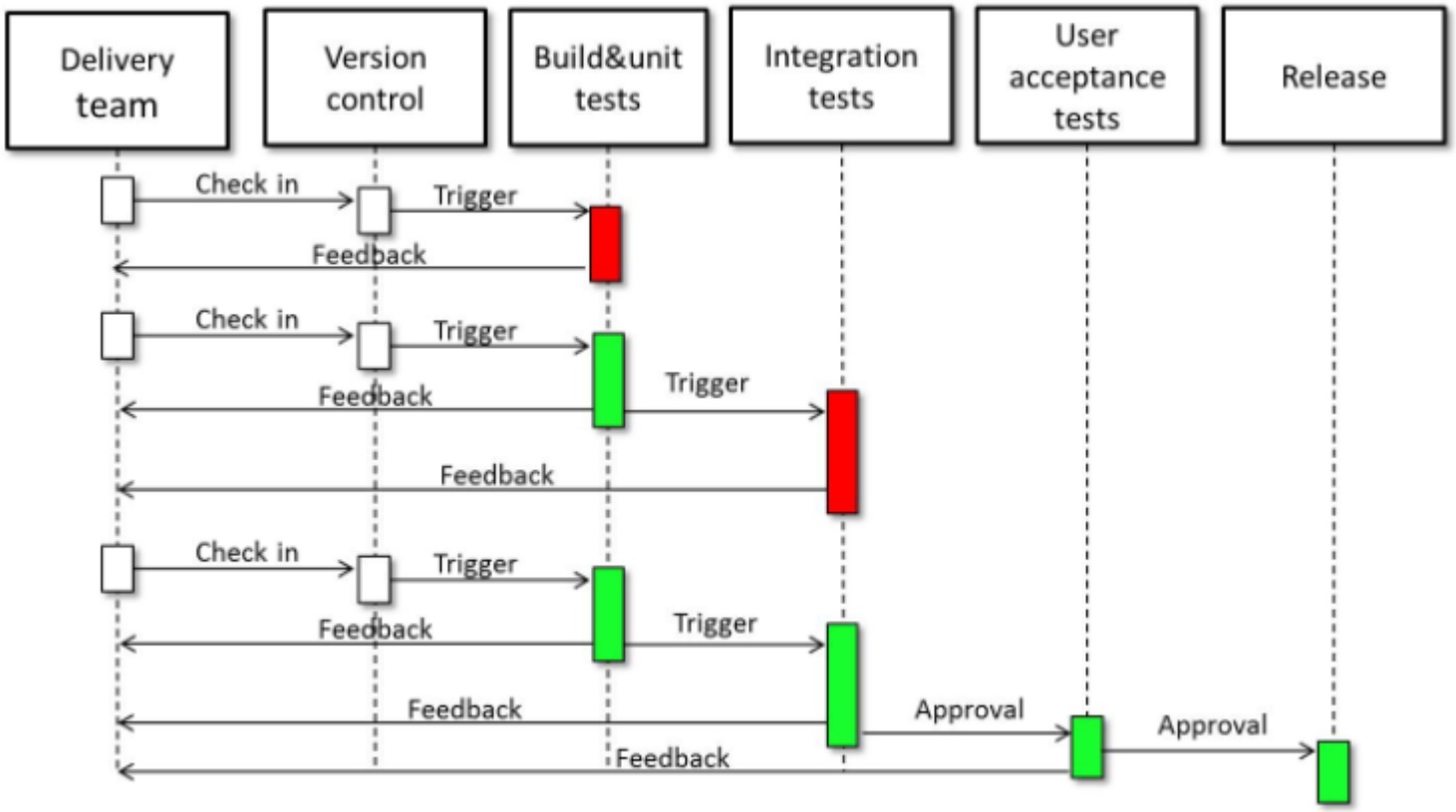
Continuous Delivery

- Iterativ
- Weiterentwicklung von Continuous Integration
- Automatisierte Test und Release Prozedur
- Defektkorrekturen schnell realisierbar
- Anforderungen:
  - Auf Commit-Ebene
- Entwicklung:
  - Erste Lieferung nicht möglich
  - Chaotisch
- Ergebnis:
  - Häufige Releases
  - Funktionierende Software nach jedem erfolgreichem Commit



Continuous Delivery: System

- Vollautomatisch
- Komplex
- Teuer
- Lange Entwicklungsdauer des Systems
- Team für Wartung nötig
  - Single Point of Failure



Vergleich

TIMEBOXING

- Iterationen
  - Wochen bis Monate
- Planung:
  - Pro Iteration
  - Horizont: nächste Iteration
- Anforderungsänderungen
  - In einer Iteration schwer
  - Durch Planung in nächster Iteration möglich

CONTINUOUS DELIVERY

- Iterationen
  - Sehr kurz
- Planung
  - Typisch keine
  - Aufgabenverteilung durch Backlog
- Anforderungsänderungen
  - Jederzeit möglich
  - Priorisierung des Backlog

Vergleich

TIMEBOXING

- Voraussetzungen:
  - Ein Team pro “Box”
  - Software ist iterativ entwickelbar
  - Auslieferung Iterativ möglich
  - Mittelgroßes bis großes Projekt
- Probleme:
  - Schwer zu koordinieren bei sehr großen Projekten
  - Nicht möglich bei kleinen Projekten
- Anwendungsbeispiele:
  - Online Anwendungen
  - Mobile Apps

CONTINUOUS DELIVERY

- Voraussetzungen:
  - Software Teams getrennt von CD Team
  - Software ist automatisch testbar
  - Auslieferung und Entwicklung kontinuierlich möglich
  - Großes Projekt – “Gigantisches” Projekt
- Probleme:
  - Minimale Software muss vorhanden sein
  - Teuer & Komplex
  - Nicht rechtfertigbar bei kleinen Projekten
- Anwendungsbeispiele:
  - Software im Finanzsektor
  - Online Anwendungen

Fragen?


ARE THERE ANY QUESTIONS? FEEL FREE TO ASK ANY-THING AT ALL.
WHY DO GHOSTS HAVE CLOTHES?
IF SOMEONE GIVES YOU A WEDGIE AT THE MOMENT YOU DIE, WILL YOU HAVE IT FOR ETERNITY?
DilbertCartoonist@gmail.com
Dilbert.com
9-22-11 ©2011 Scott Adams, Inc./Dist. by Universal Uclick

Vielen Dank

Benjamin Projdakov

Vergleich:

A: The Timeboxing Process Model for Iterative Software Development

B: Release Management with Continuous Delivery: A Case Study