

# *The Eclipse Way*



Frage:

Wie wird das Eclipse-SDK  
entwickelt?



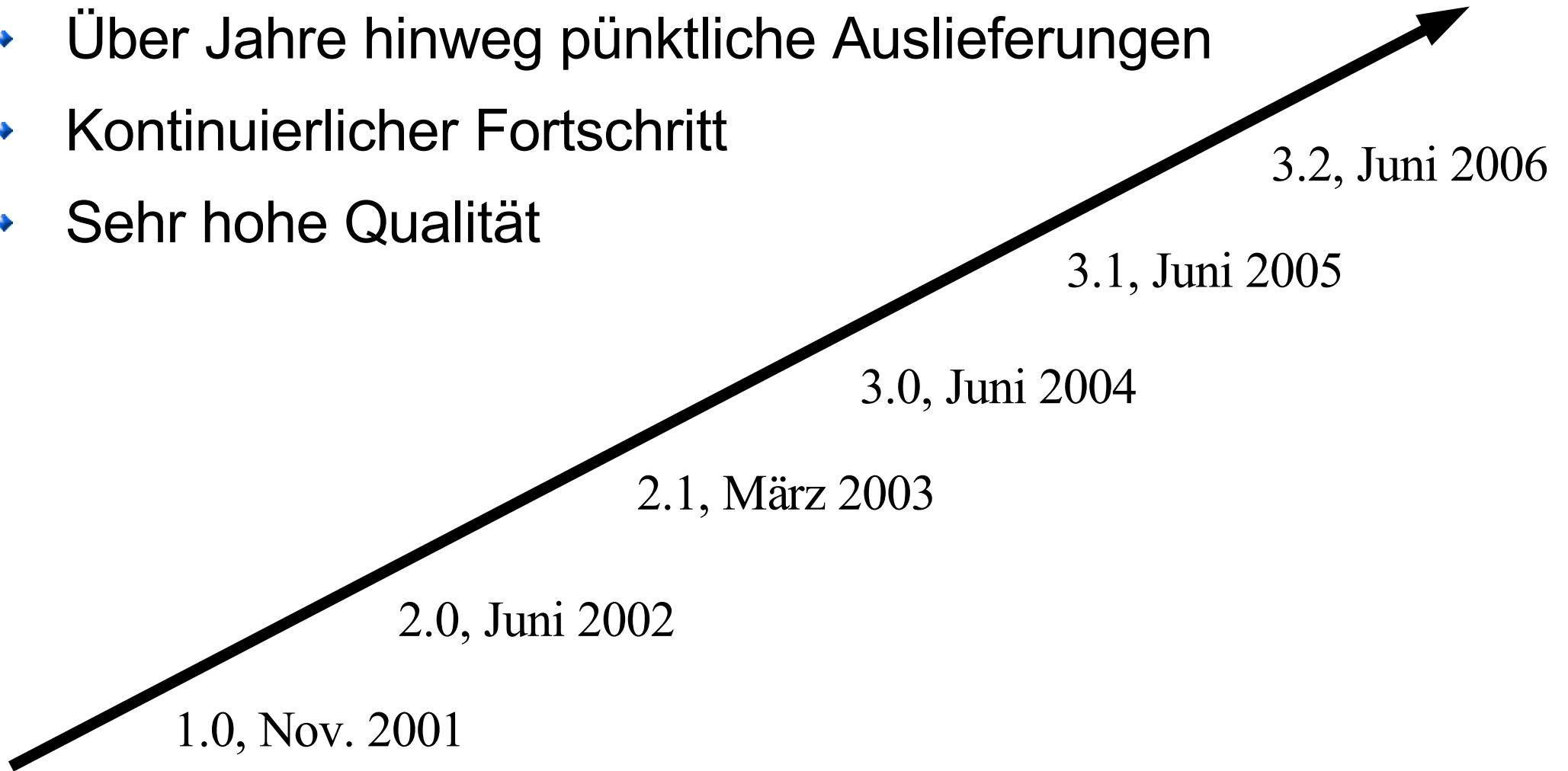
# Überblick

- ♦ Werte und Techniken
- ♦ Der Rhythmus des Projekts
- ♦ Die Schluss-Phase



# Die Stärken

- ♦ Über Jahre hinweg pünktliche Auslieferungen
- ♦ Kontinuierlicher Fortschritt
- ♦ Sehr hohe Qualität



# ***Erfolgsfaktoren***

- ◆ Sehr gute Entwickler
- ◆ Langjährige Erfahrung mit dem Bau von IDEs
- ◆ Kleine Teams
  
- ◆ Vor allem: Agiler Entwicklungsprozess



# *Die grundlegende Annahme*

- ◆ “People and interactions over processes and tools”

Agiles Manifest

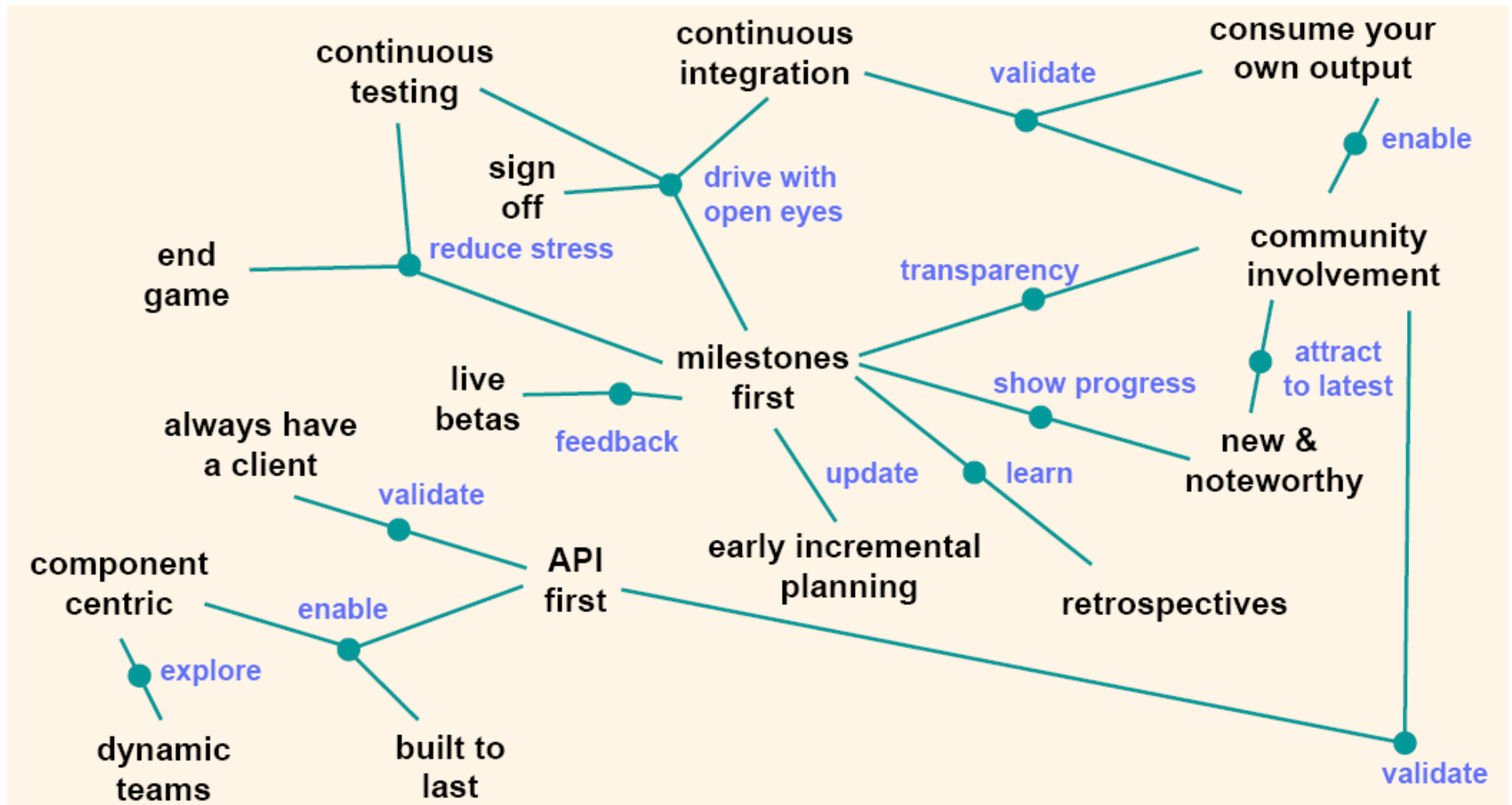


# Werte

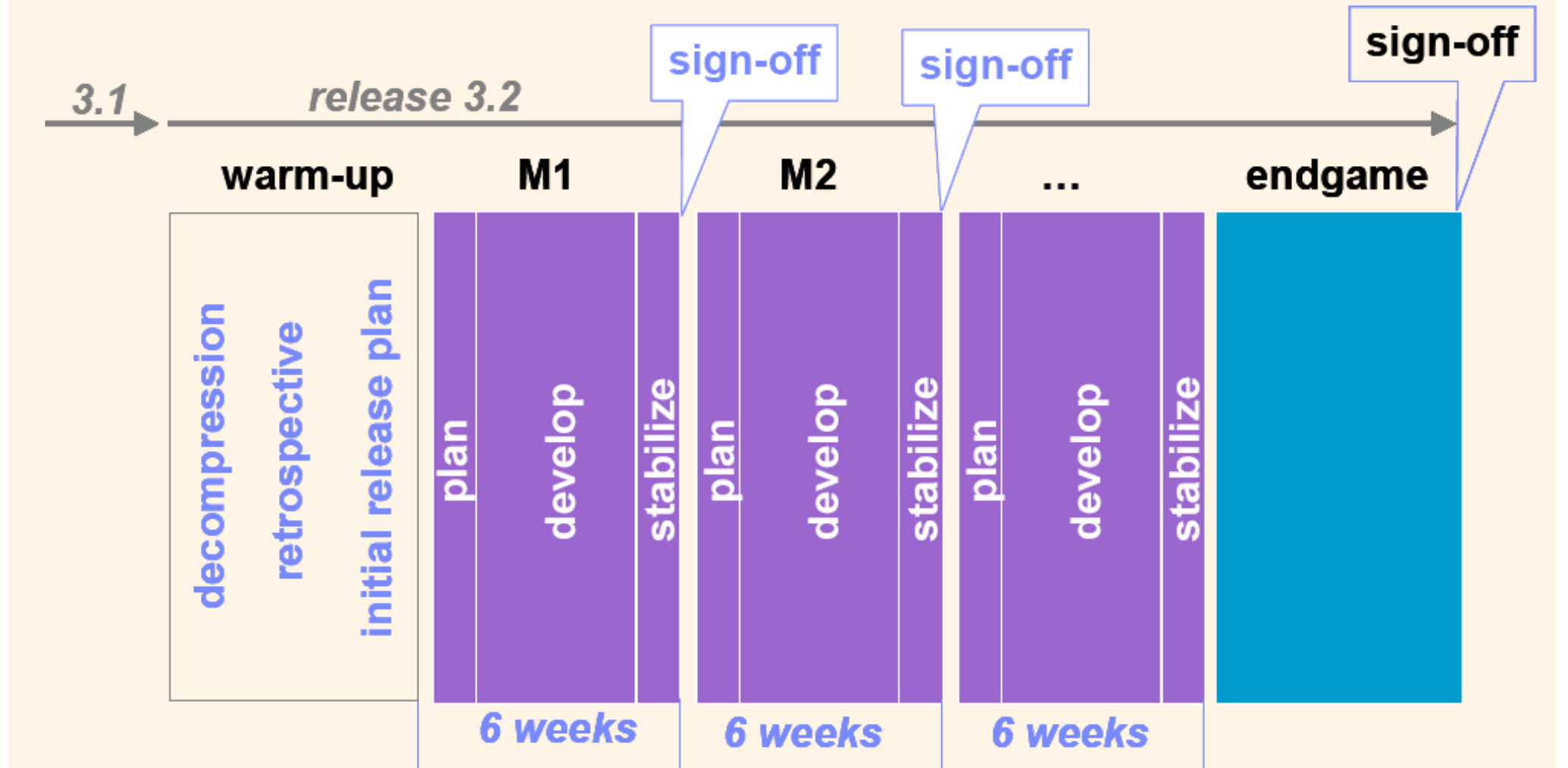
- ◆ Software ausliefern
- ◆ Vorhersagbarkeit
  - ◆ Pünktliche Auslieferungen
- ◆ Transparenz
  - ◆ Keine Geheimnisse über den Projektstatus
- ◆ Feedback



# Techniken



# Der Rhythmus des Projekts



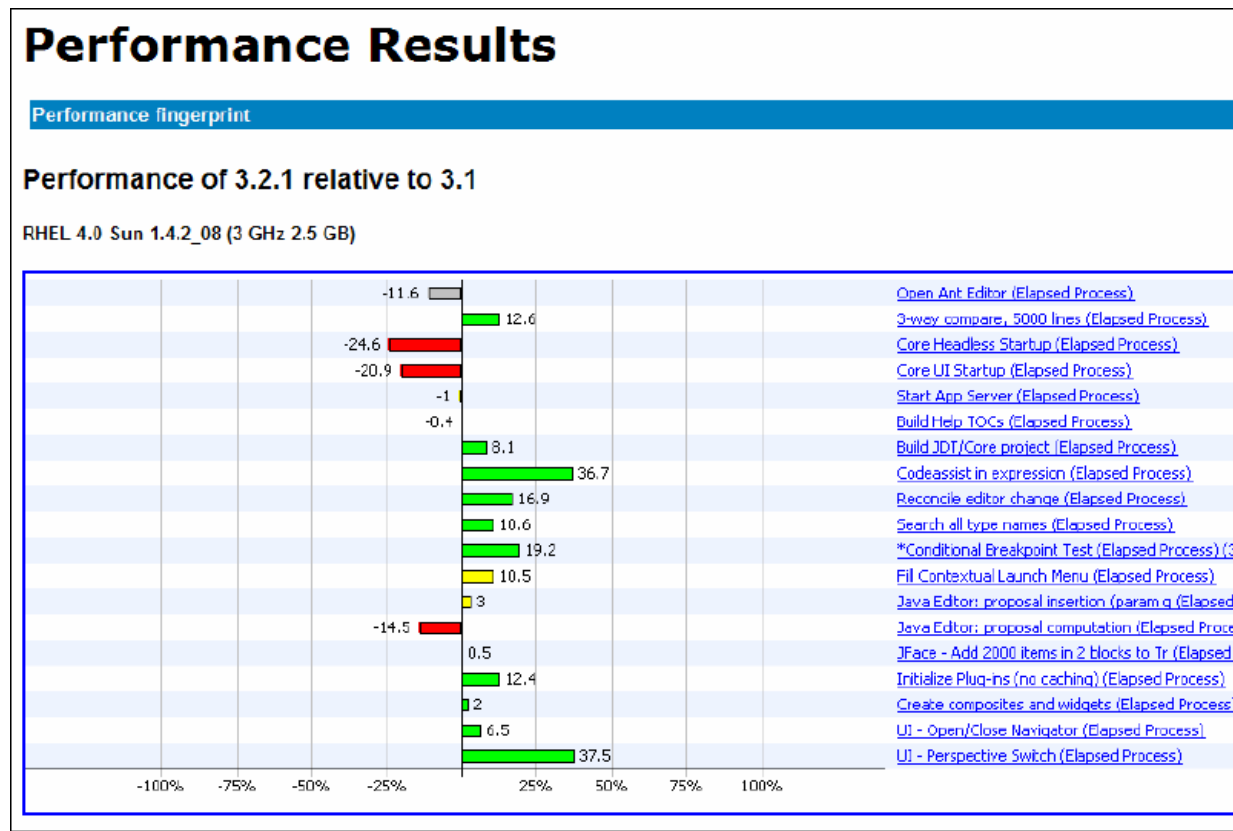
# *Continuous Builds & Tests*

- ◆ Nightly Build
  - ◆ Alle Unit-Tests ausführen (> 30.000, Dauer ca. 12h)
  - ◆ Ergebnisse publizieren
- ◆ Integration Build
  - ◆ 1x pro Woche
  - ◆ Alle Unit-Tests ausführen
- ◆ Milestone Build
  - ◆ Alle 6 Wochen
  - ◆ Wie ein kleines Release



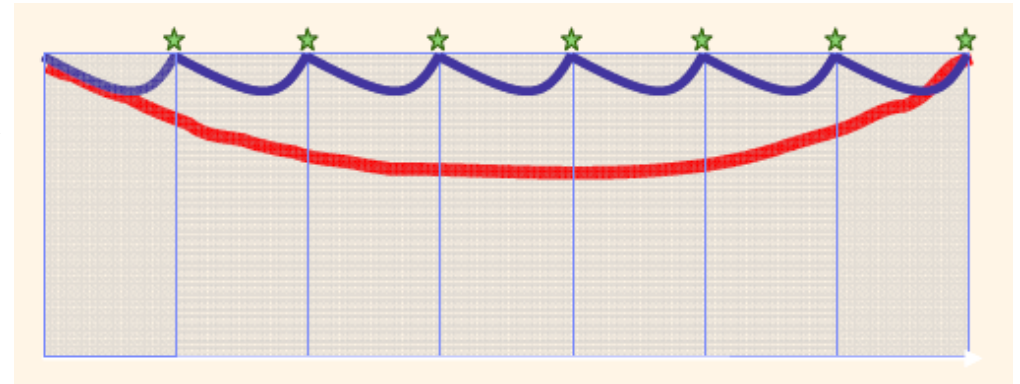
# Performance-Tests

- ▶ Neben den umfangreichen Unit-Tests werden automatisierte Performance-Tests durchgeführt



# Milestones

- ▶ Alle 6 Wochen: Milestone-Build
- ▶ Reduziert Stress
- ▶ Sorgt für schnelles Feedback



- ▶ Feedback gibt es nicht automatisch
- ▶ New and Noteworthy → Werbung
- ▶ Neue Features



# *Stabilization*

- ◆ Montag: warm-up
- ◆ Dienstag: validate
- ◆ Mittwoch: test
- ◆ Donnerstag: fix
- ◆ Freitag: verify, go / no go, sign-off



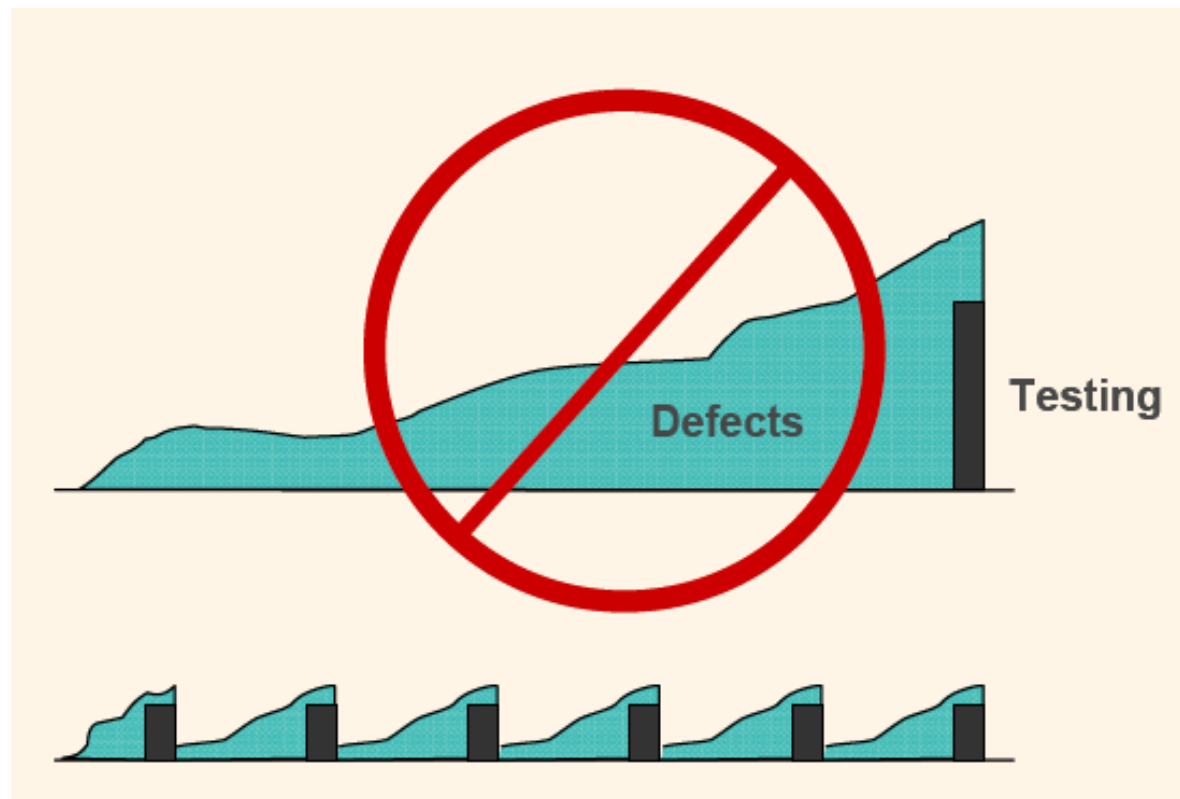
# *Live Betas*

- ♦ Projekt befindet sich immer im Beta-Status
- ♦ “Project health is stronger than quality”
  - ♦ Qualität zu einem bestimmten Zeitpunkt ist nicht ausreichend!
- ♦ Kontinuierlich...
  - ♦ ... verwendbare Software
  - ♦ ... interessante Neuerungen
  - ♦ ... zuhören und Einfluss ermöglichen (motiviert Feedback)
- ♦ “Consume your own Output”

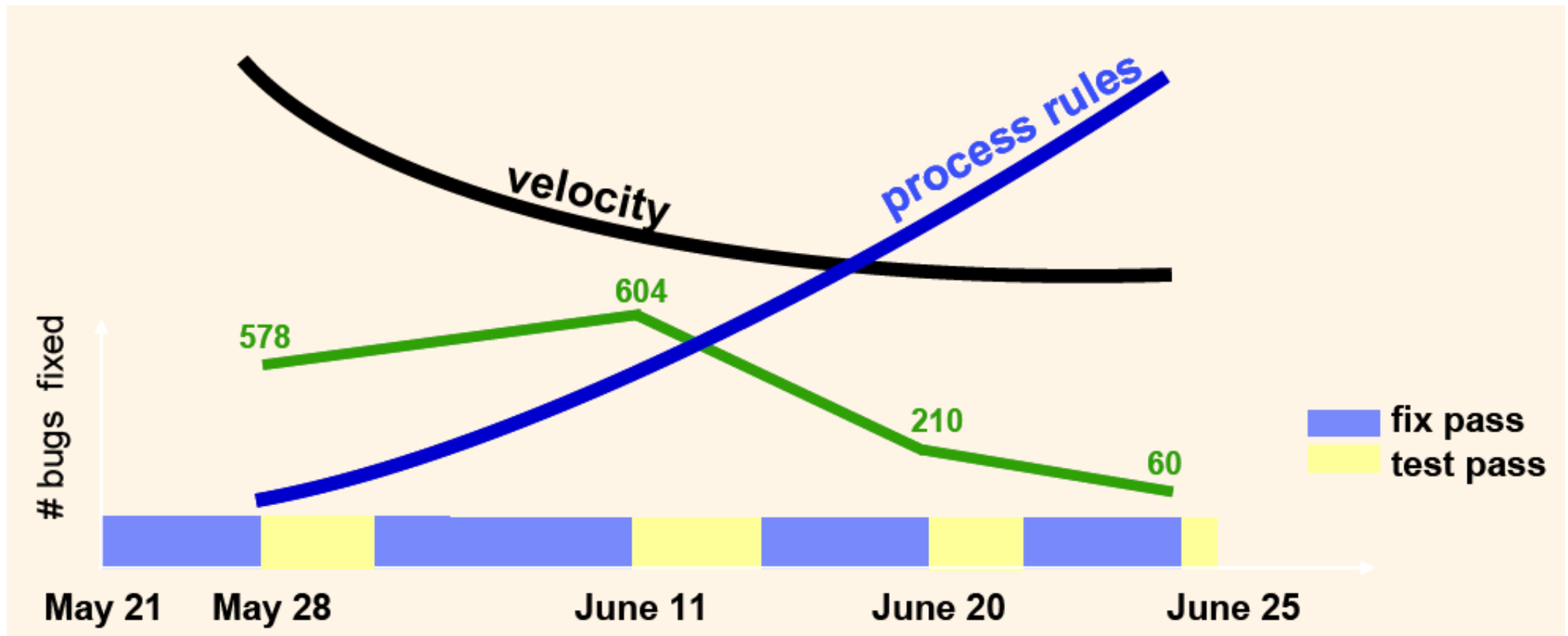


# Endgame

- Vom Beta-Status zum Release
- Wechsel zwischen festen Test- und Bugfix-Phasen



# Endgame



# *Continuous...*

- ◆ ... transparent planning
- ◆ ... design / refactoring
- ◆ ... integration
- ◆ ... testing
- ◆ ... listening
- ◆ ... demos
- ◆ ... consumption of our own output
- ◆ ... feedback
- ◆ ... learning
- ◆ ► continuous health

