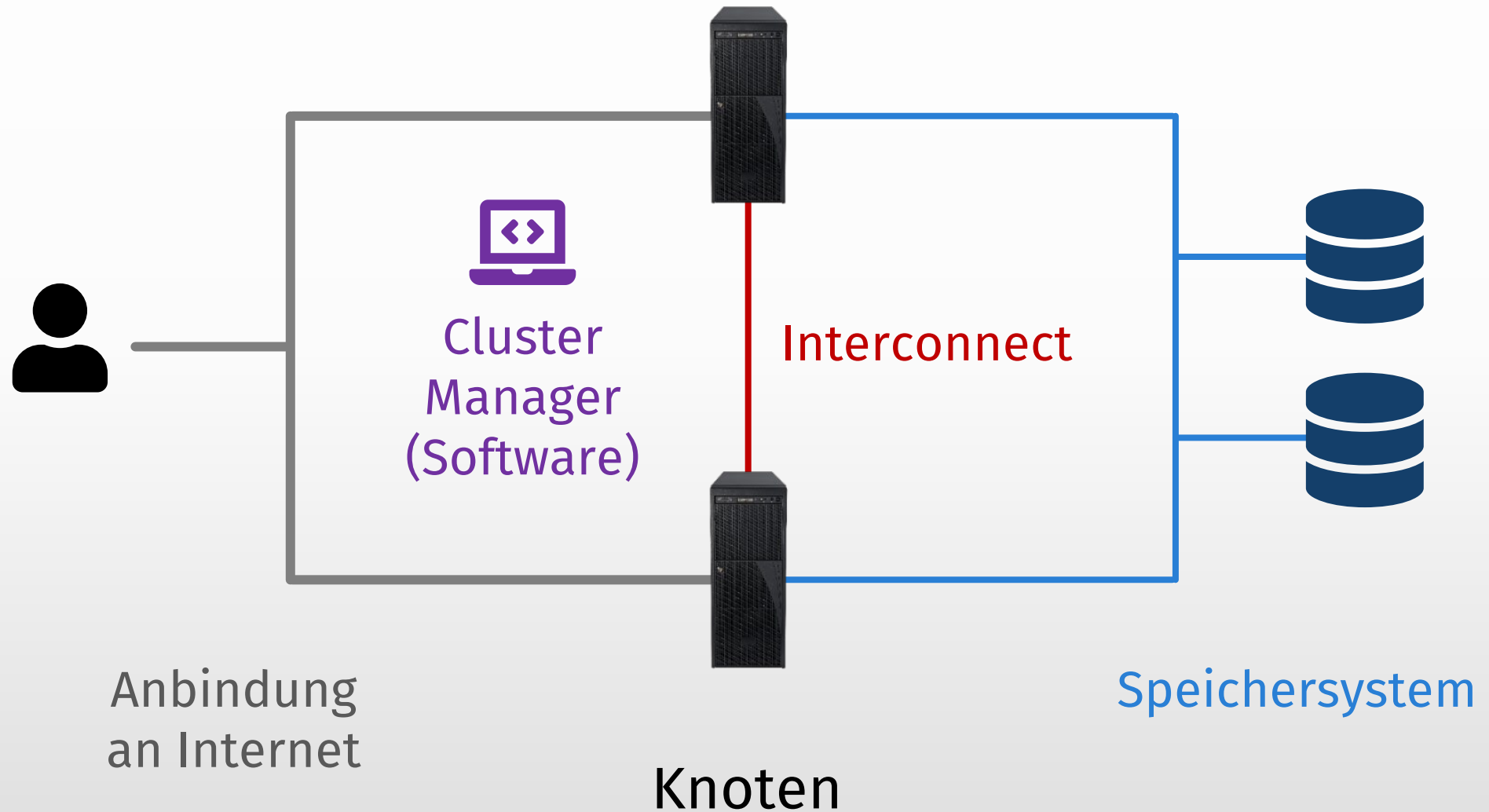


**Application Performance Management**

# **Performance Testing im Web**

Michael Faes

# Rückblick: High Availability & Failover



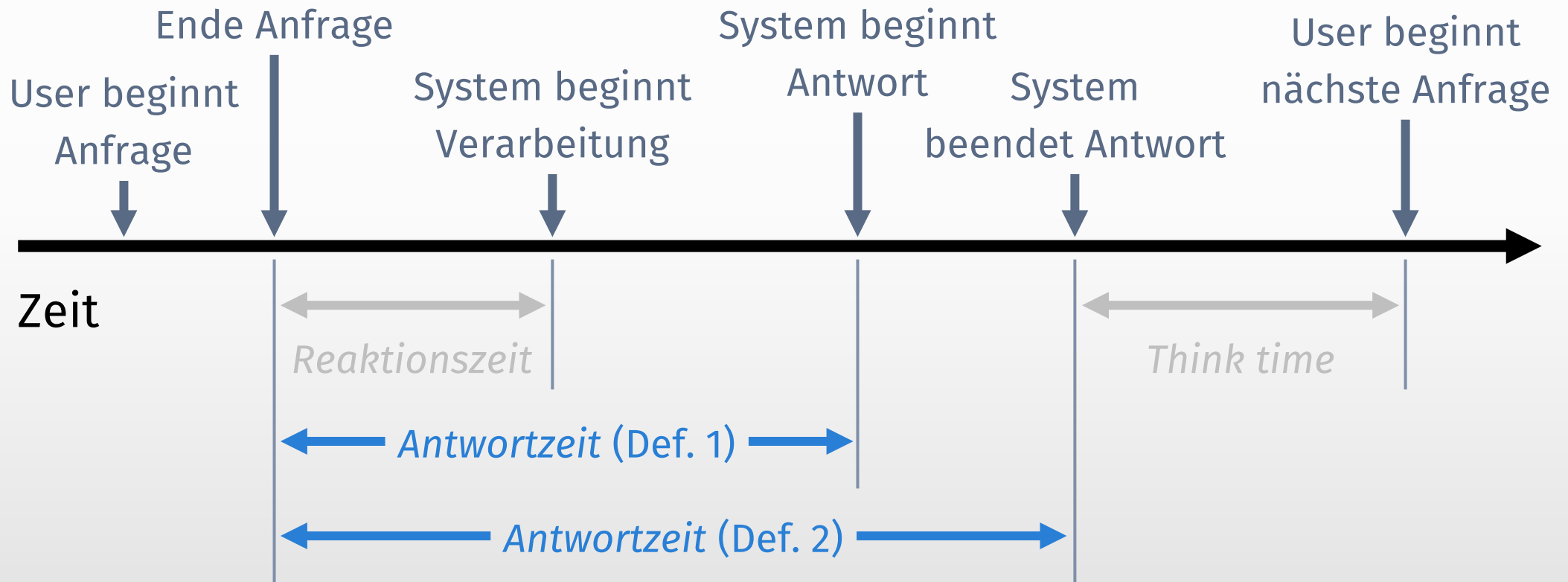
# Übersicht

1. Rückblick High Availability
2. Performance messen im Web
  - Endbenutzerantwortzeit
  - Aktives & passives Monitoring
  - Workload Characterization
  - Apache JMeter
3. Performance Testing
  - Arten von Performance Tests
  - Last- & Stressverhalten von Apps
4. Übung

# **Performance Messen im Web**

# Rückblick: Antwortzeit

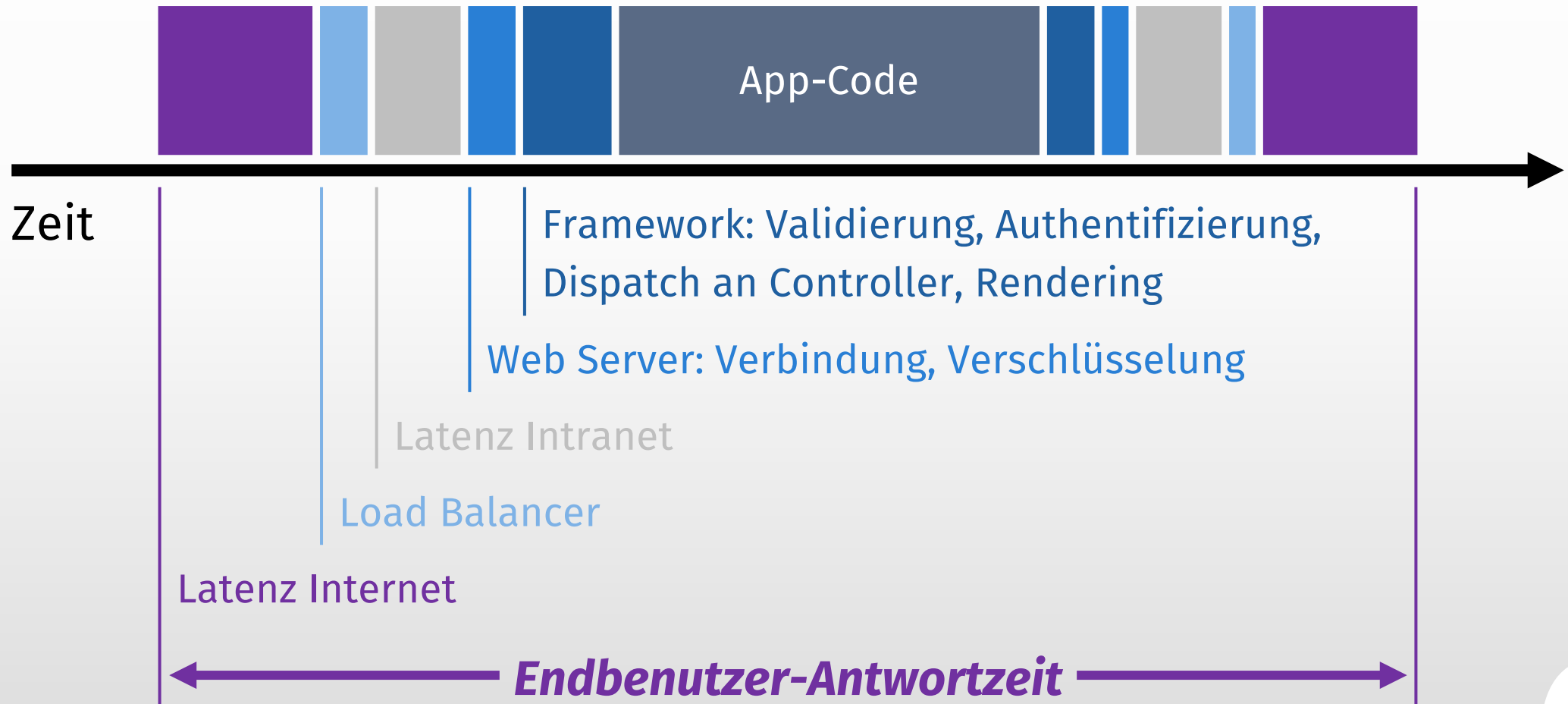
Wichtigste Metrik aus Sicht des Benutzers: *Antwortzeit*



Definitionen sind alle **aus der Sicht des Systems!**

# Systemgrenze und Zeitmessung

Wo ist Grenze des «Systems»? Wo messen wir Antwortzeit?



# Endbenutzer-Antwortzeit

Kann objektiv gemessen werden, aber wird subjektiv wahrgenommen.

*Die empfundene durchschnittliche Antwortzeit entspricht nicht dem Durchschnitt, sondern dem 0.9-Quantil, das heisst dem Wert, der grösser ist als 90% aller beobachteten Antwortzeiten.*

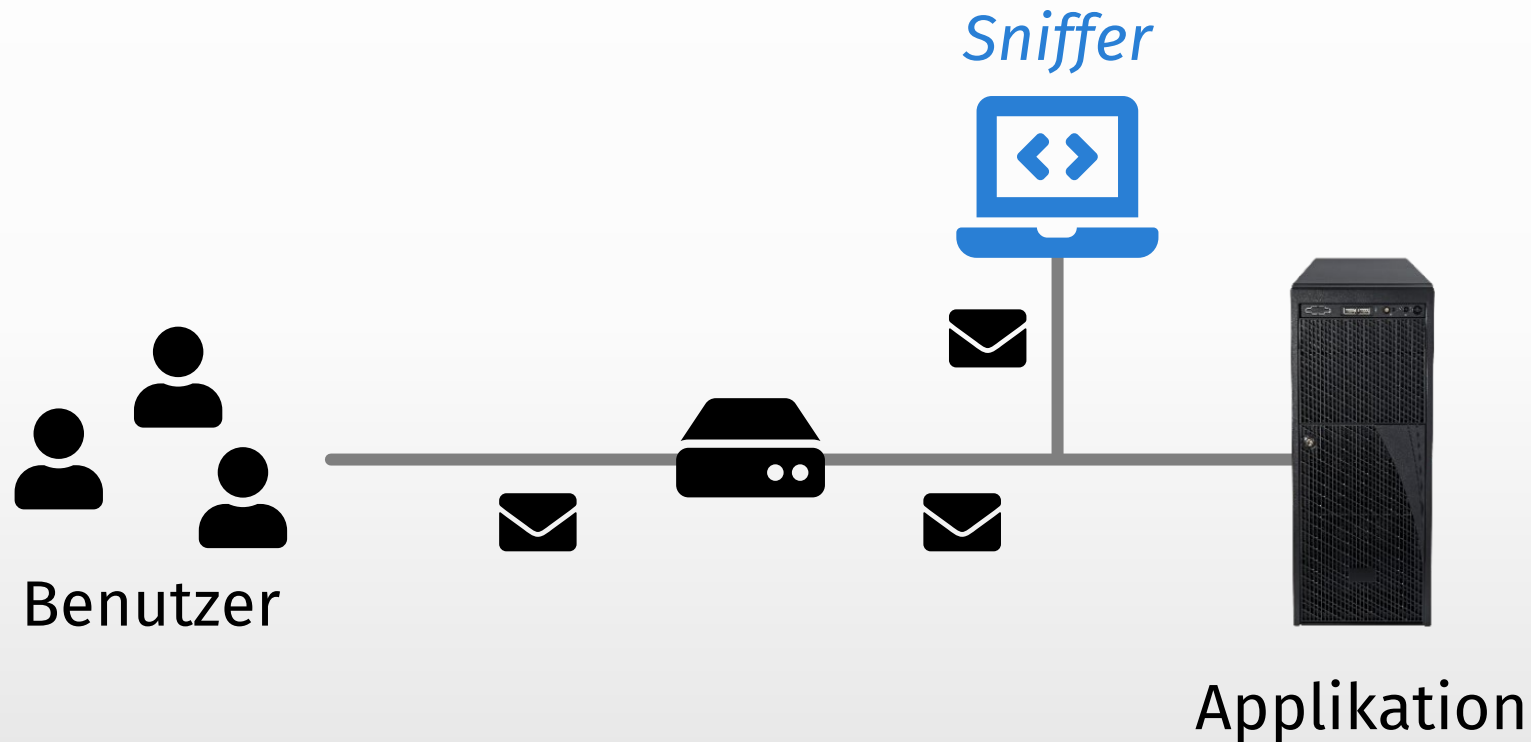
— Arnold O. Allen, Introduction to Computer Performance Analysis with Mathematica (Academic Press, San Diego, CA, 1994)

## Wie messen / abschätzen?

- Passives Monitoring
- Aktives Monitoring

# Passives Monitoring

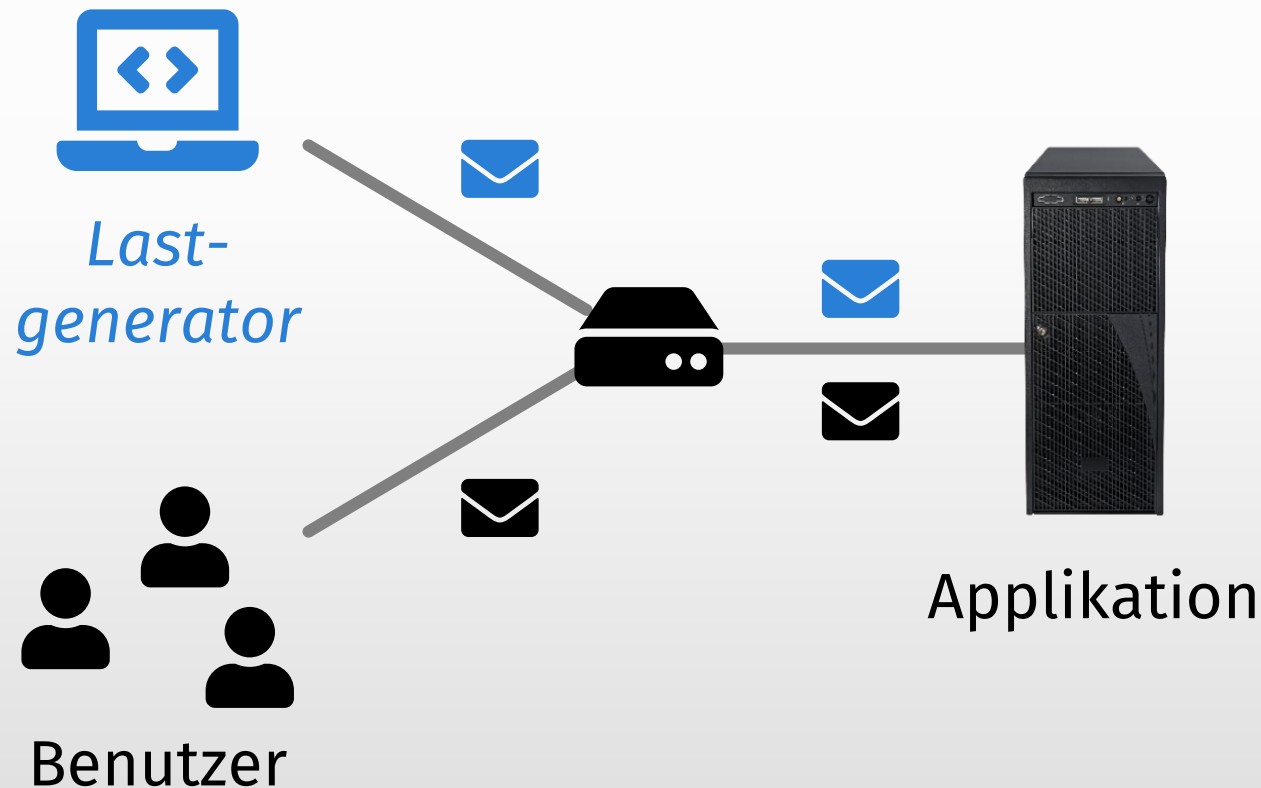
Beobachten von Benutzer-Anfragen, z. B. mit Netzwerk-Sniffer





# Aktives Monitoring

Ein *Lastgenerator* erzeugt synthetische Anfragen, die «typisches» Benutzerverhalten simulieren:



# Aktives vs. passives Monitoring

## Aktiv

- Misst vollständige Endbenutzer-Antwortzeit (...)
- Belastet System mit zusätzlichen Anfragen
- Keine echten Users nötig, flexibler Einsatz in Entwicklung
- Benötigt Modell für «typisches» Benutzerverhalten

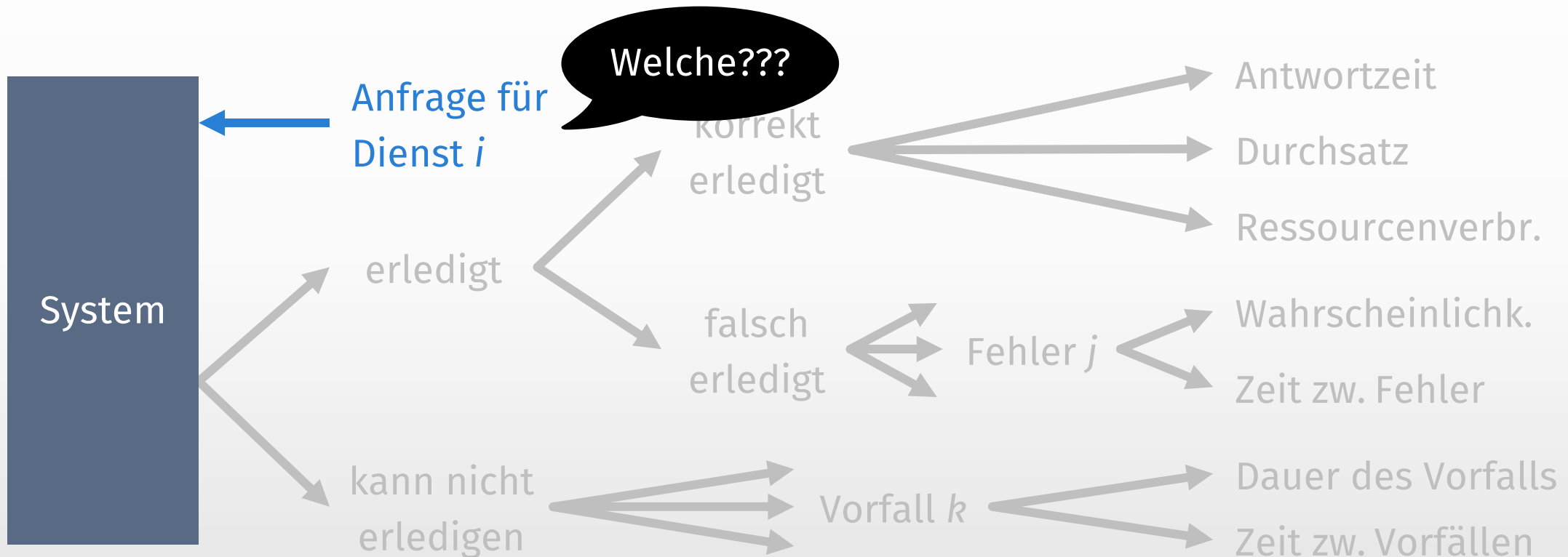
## Passiv

- Misst nur Teil der Endbenutzer-Antwortzeit
- Keine Zusatzbelastung des Systems
- Benötigt echte Users und laufendes Produktiv-System
- Kein Modell nötig, beobachtet echtes Benutzerverhalten

→ *Workload Characterization*

# Auswahl des Workloads

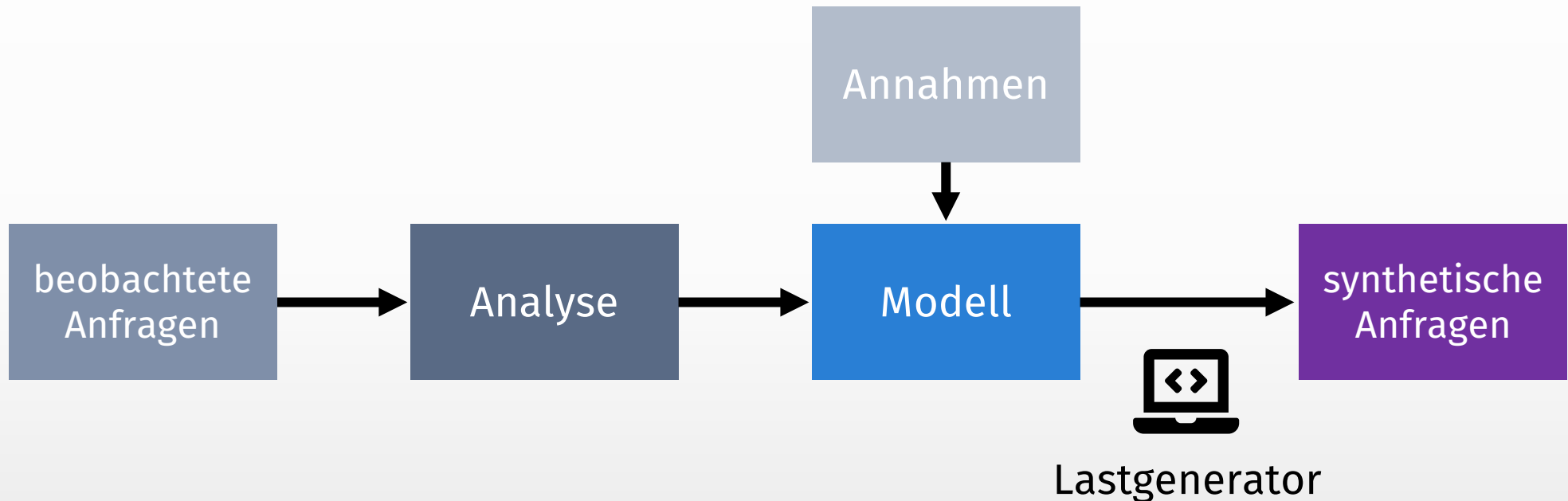
Einer der wichtigsten Teile von Performance-Analyse: der *Workload*.



Ausgewählte/beobachtete Last hat grossen Einfluss auf Resultate.

# Workload Selection / Characterization

Aktives Monitoring braucht *Modell* für Benutzerverhalten.



**Beispiel-Modell** (sehr einfach):

Mach GET-Anfrage an /shop. Warte 30 Sek. Wiederhole.

Workload besteht z. B. aus 50 solcher Benutzer mit 20 Wiederholungen.

Think time!

# Überlegungen bei der Auswahl und Charakterisierung von Workloads:

## 1. Komplexität

- Häufigste Anfrage
- Mischung von Anfragen, mit definierter Häufigkeit
- *Sequenzen von Anfragen*, inklusive Think times
- ...



Vor-/  
Nachteile?

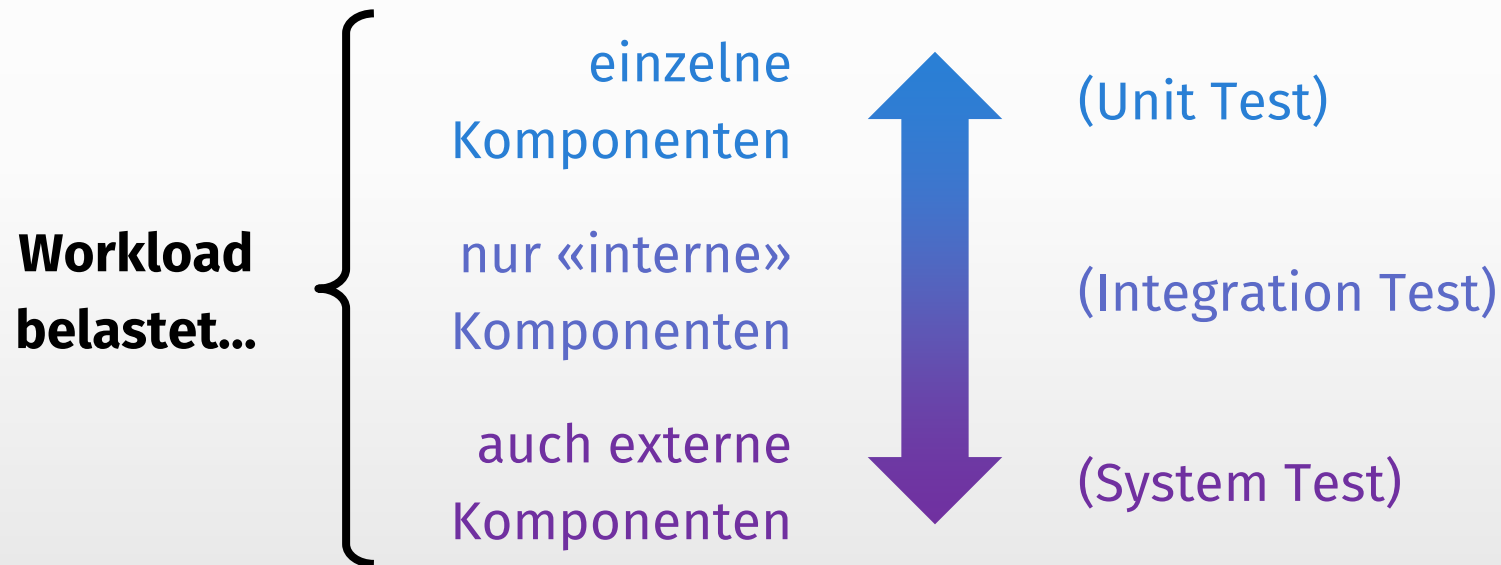
**2. Aktualität:** Workload soll *aktuellem* Benutzerverhalten entsprechen.

Herausforderungen:

- Können Verhalten immer nur für bestimmte Situation beobachten.  
Kann abhängig sein von Tageszeit, Jahreszeit, Wetter, Ereignissen, ...
- Verhalten kann sogar abhängig sein von App-Performance!  
*Verhalten beeinflusst Performance und umgekehrt!*

**3. Variabilität:** Variables Benutzerverhalten kann gut durch *Zufall* simuliert werden. Aber Resultate sollen auch möglichst *reproduzierbar* sein! Viel Zufall führt zu weniger Reproduzierbarkeit...

**4. Scope:** Wie bei funktionalen Tests gibt es verschiedene Grössenordnungen zum «Testen» von Performance:



**5. Last-Level:** System unter «niedriger», «erwarteter» oder «hoher» Last testen? Oder sogar zum Zusammenbruch bringen? → **Zweck!**

# Apache JMeter

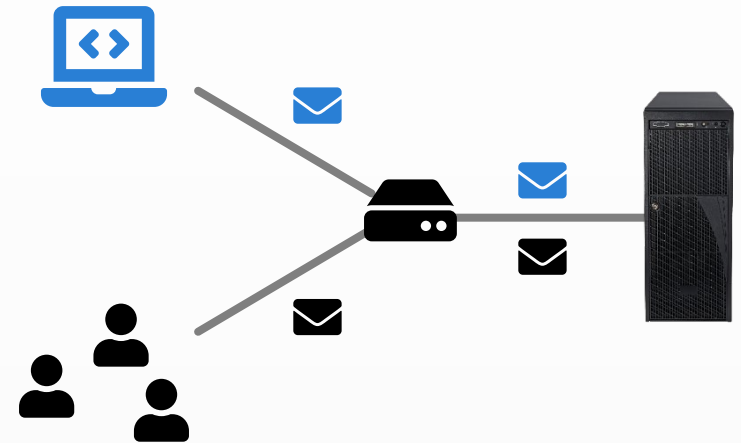
*Apache JMeter*: Lastgenerator-Software.  
Generiert Anfragen und misst  
Antwortzeit & Durchsatz

Unterstützung für Web-Apps, Web-  
Dienste, Datenbanken, Verbindungen

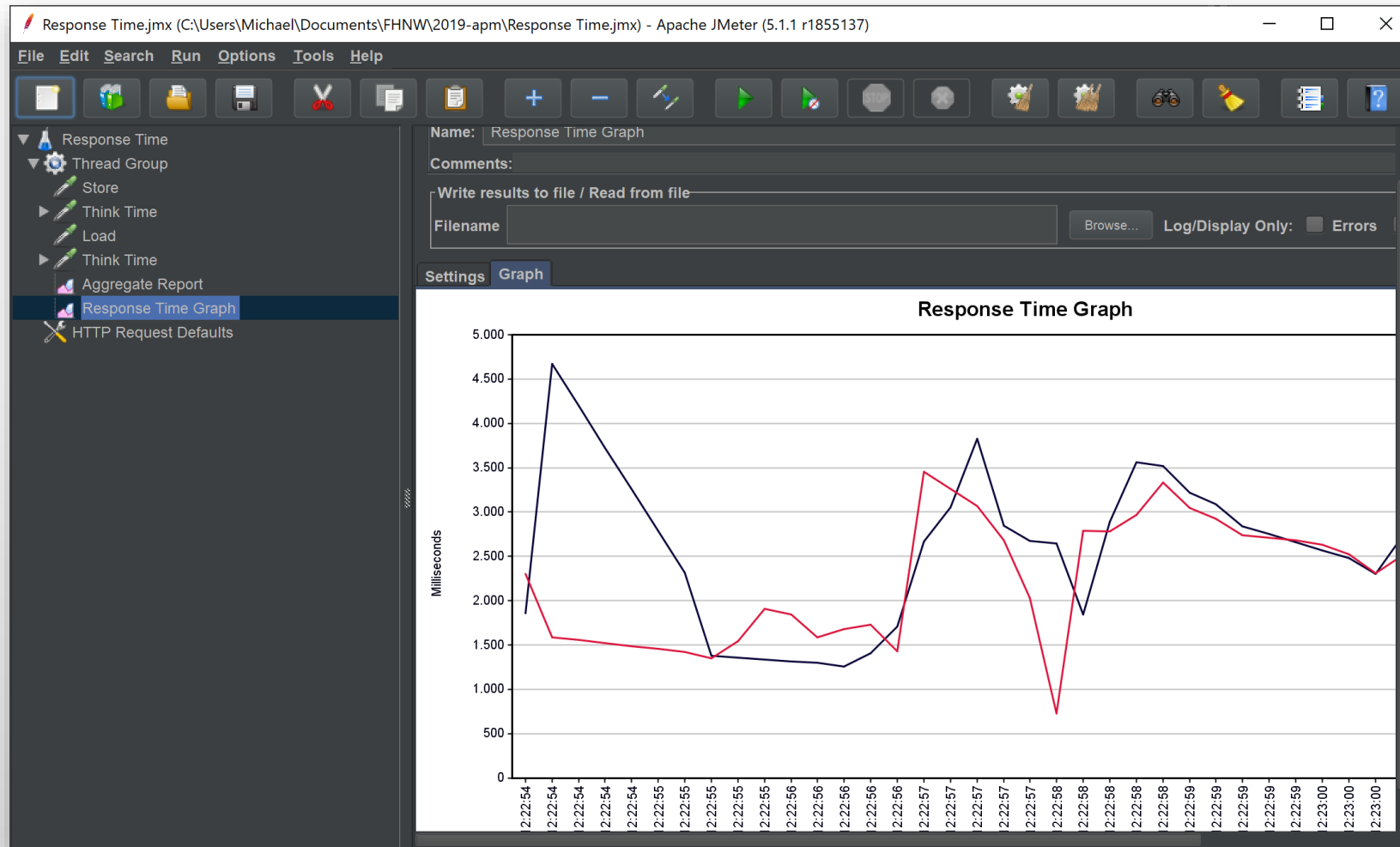
- Protokolle: HTTP(S), REST, SOAP, FTP, LDAP, JDBC, JMS, SMTP, IMAP, TCP, ...

Möglichkeiten für Workload-Definition:

- GUI
- Scripting
- Browser-Recording



# Verwendung von JMeter (Live)





# Performance Testing

# Performance «testen»

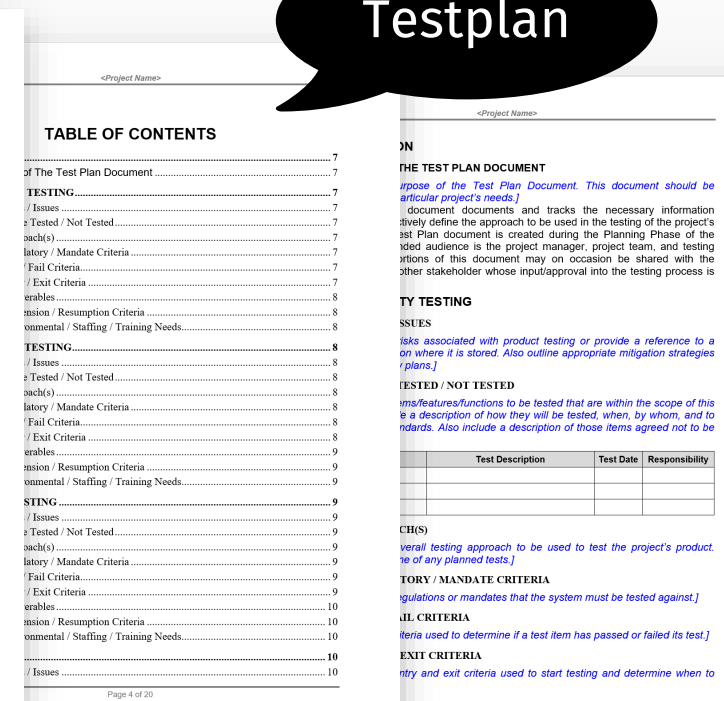
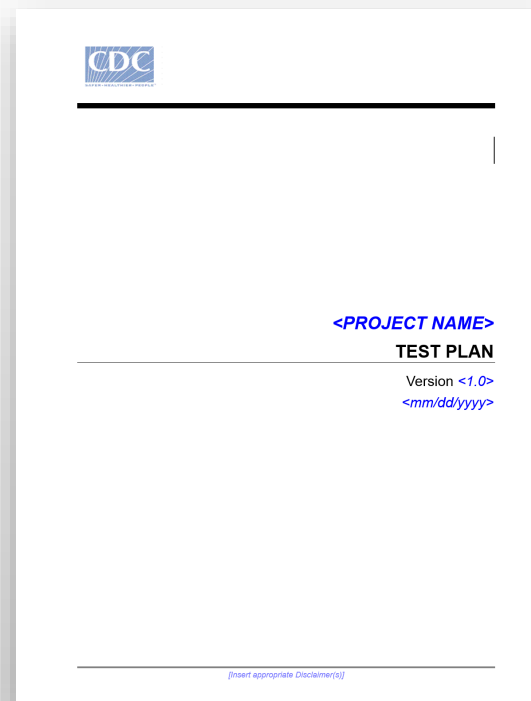
Haben gesehen, wie man Performance *misst* und *auswertet*

- Erste Semesterhälfte: einzelne Code-Stücke, *Single-Node-Apps*
- Heute: (verteilte) Web-Apps

**Nächster Schritt:** *Performance testen*. Unterschied zu *Messung*?

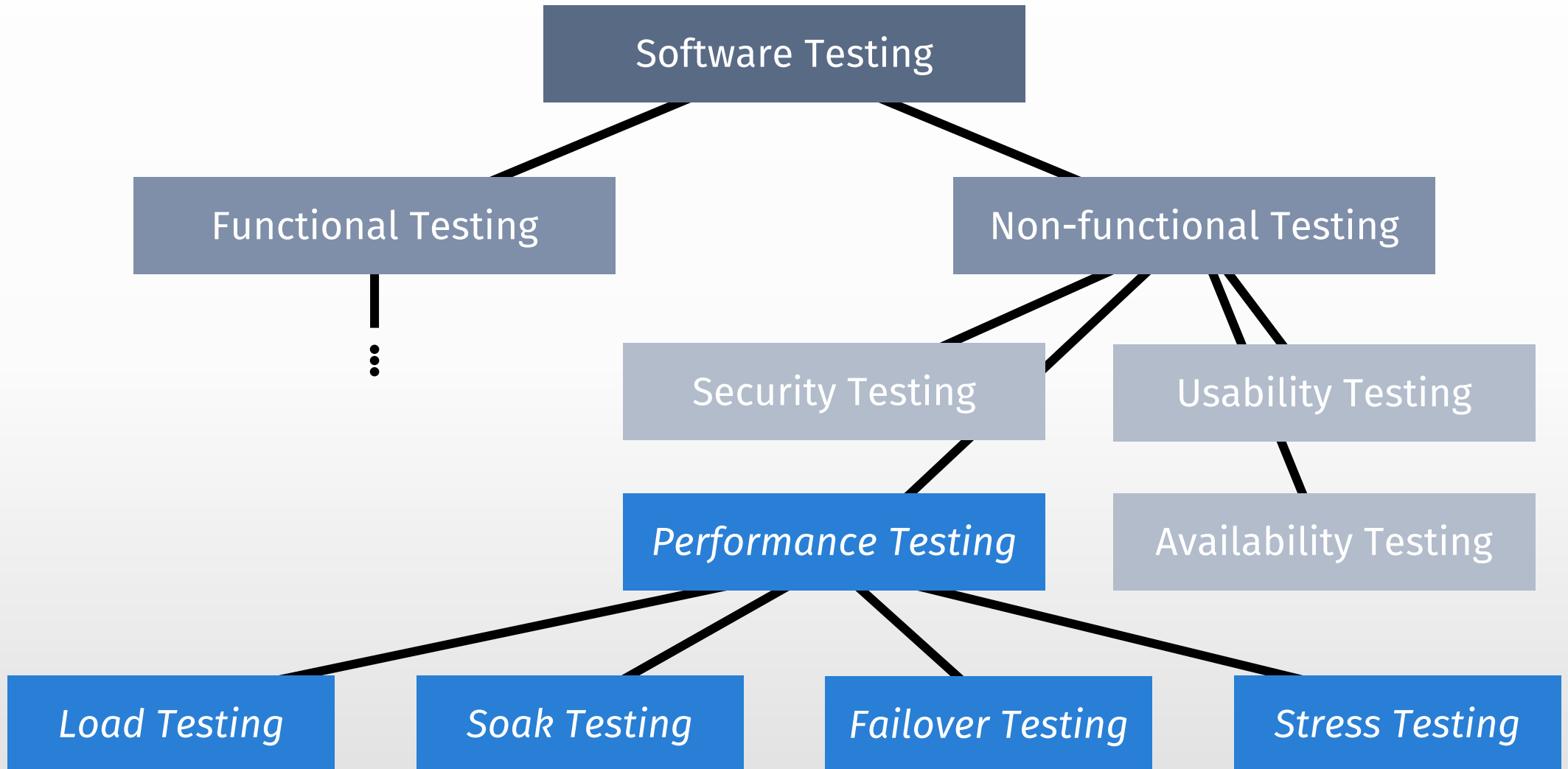
Engineering-Ansatz!

- Requirements-Analyse
- Planung
- Entwicklung von Test-Suite
- Dokumentation
- Reporting
- ...



Testplan

# Performance Testing



# Arten von Performance-Tests

## **Load Test** (*Lasttest*)

Analyse der Performance unter einer *konstanten, definierten* Last

- Z. B. Antwortzeiten, Durchsatz, Ressourcenverbrauch

**Ziel:** Sicherstellen, dass Leistung unter erwarteten Last ausreichend ist

Wird z. B. vor dem Live-Schalten einer App oder vor erwarteter Zunahme der Last gemacht

## **Soak Test** (*Dauerlasttest*)

Untersuchen der Performance *über längeren Zeitraum hinweg*

- Z. B. 12 Stunden oder mehrere Tage

**Ziel:** Identifizieren von Memory Leaks oder anderen Anomalitäten

## ***Failover Test***

Lasttest bei (manuell verursachtem) Ausfall von Systemkomponenten

### **Ziele:**

- Sicherstellen, dass Failover zuverlässig funktioniert
- Analyse von Performance während/nach Failover (z. B. zusätzlicher Ressourcenverbrauch)

## ***Stress Test***

Schrittweises Erhöhen der Last, solange bis System instabil wird oder komplett ausfällt

**Ziel:** Verstehen des Verhaltens des Systems unter extremer Last

# Stressverhalten von Applikationen

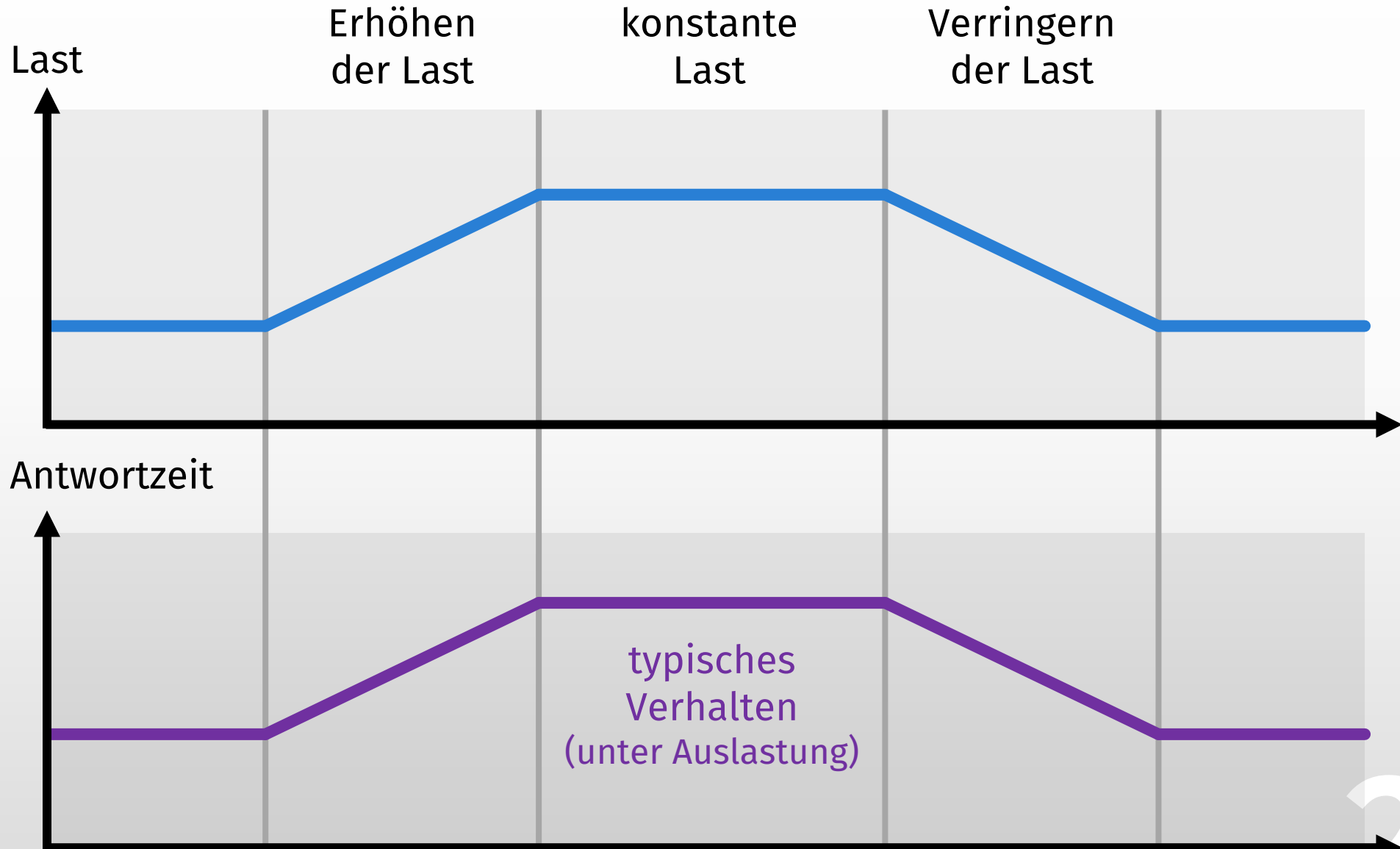
Konkrete Fragen beim Stresstesten:

- Wie ändert sich Antwortzeit-Verhalten?
- Wie gross ist maximale Last, ohne dass System unzuverlässig wird?
- Zeigt System unerwartetes Verhalten?

**Stressverhalten:** Welche Effekte können auftreten?

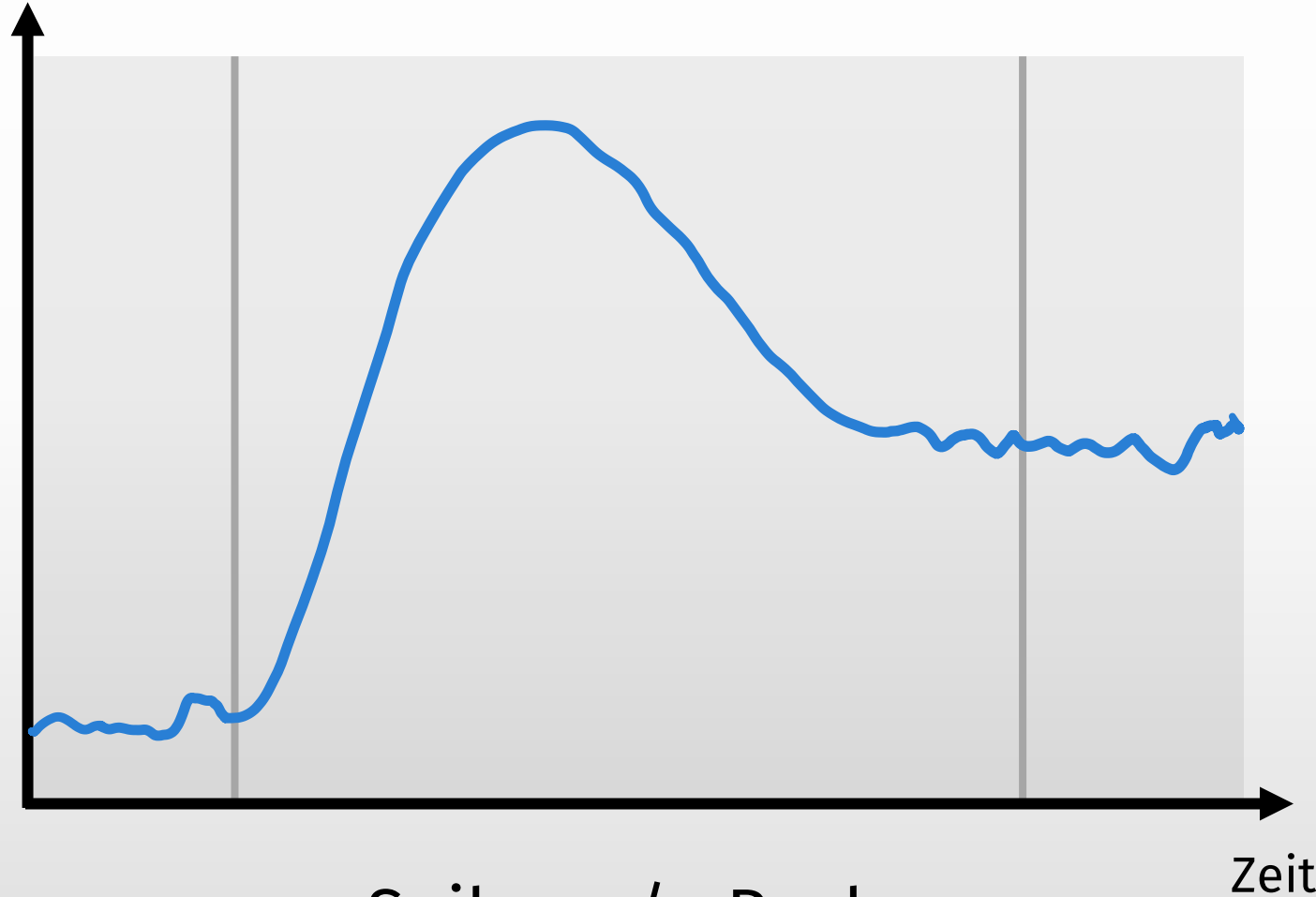
- Einzelne Aussetzer → Einfluss auf Verfügbarkeit (SLA!)
- Totalausfall
- Datenverlust/-Inkonsistenz
- Failover
- Kettenreaktion durch Failover

# Allgemein: Lastverhalten



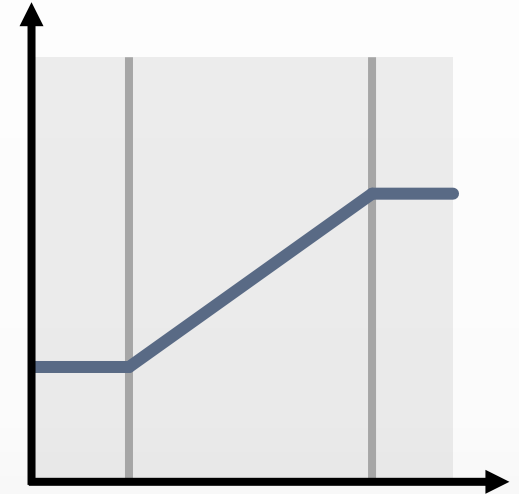
# Erhöhen der Last

Antwortzeit



«Spikes» / «Peaks»

Zeit

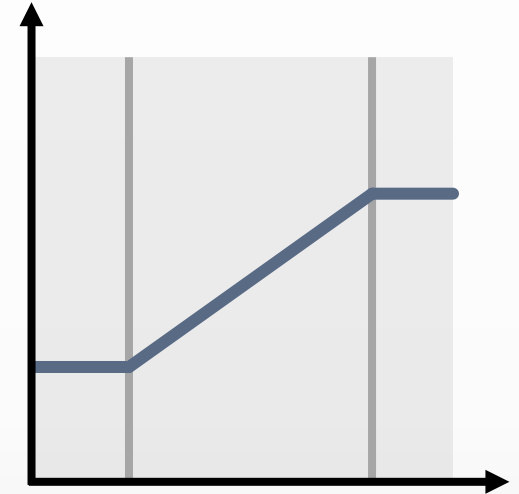
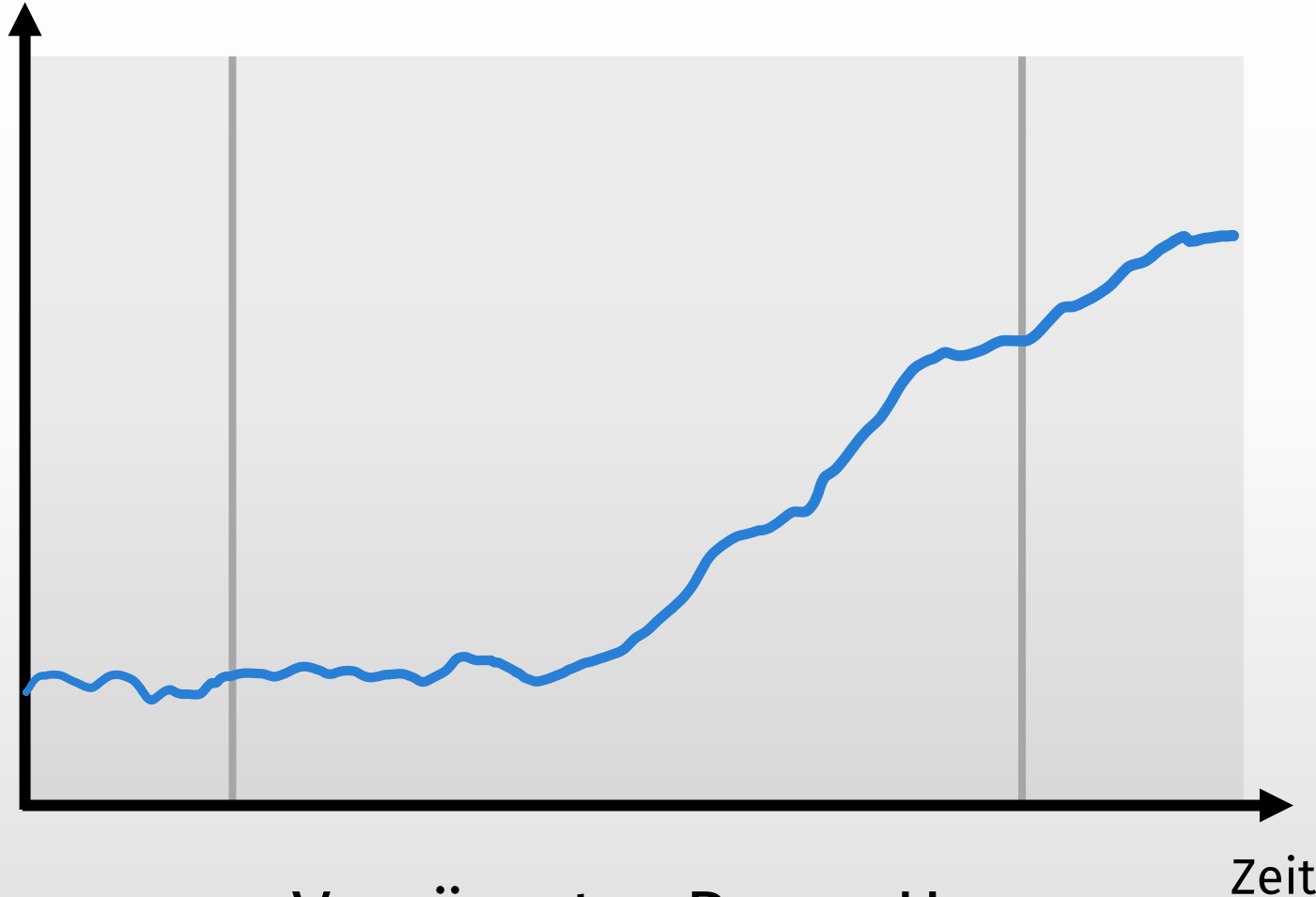


typisches  
Verhalten



# Erhöhen der Last

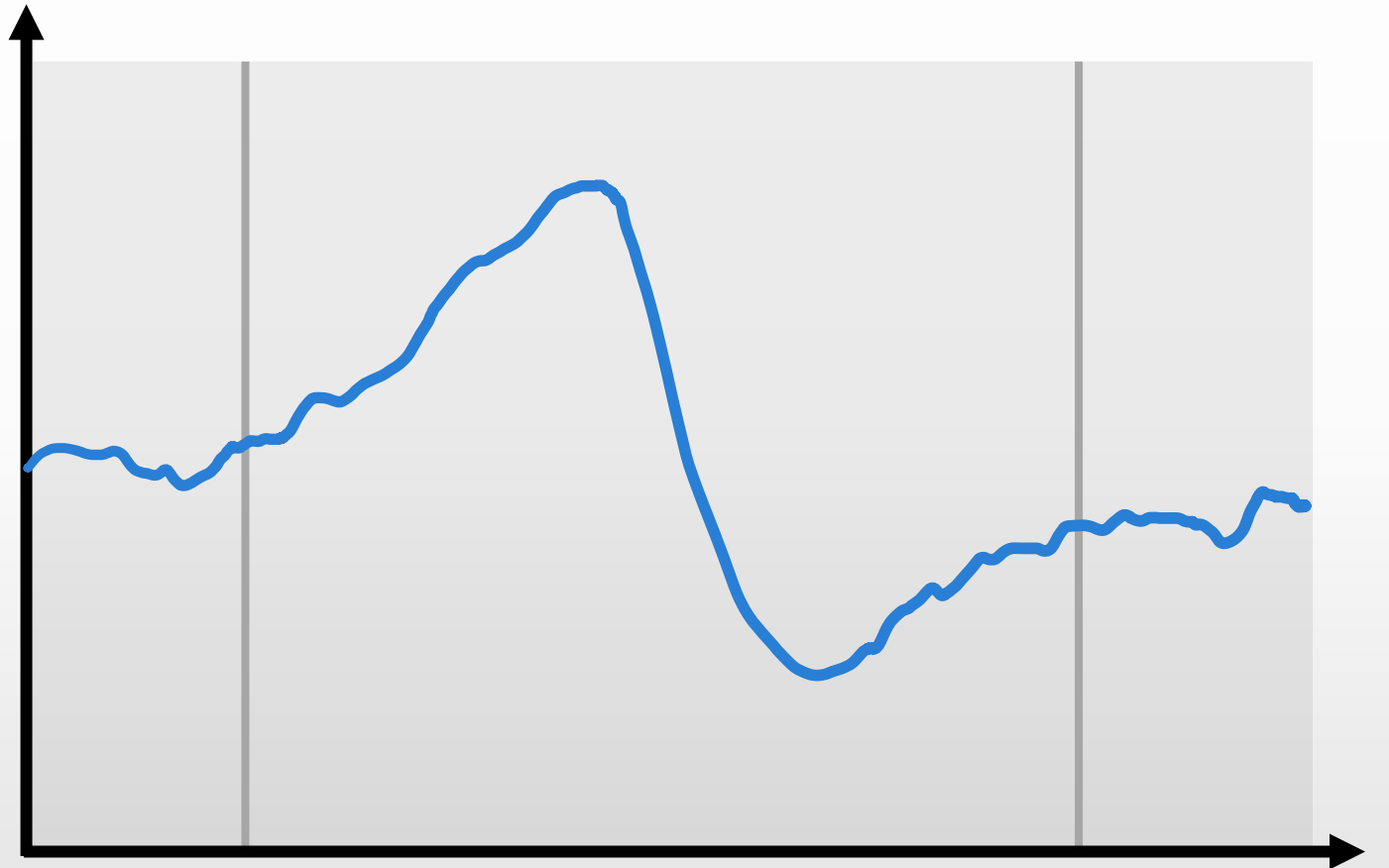
Antwortzeit



typisches  
Verhalten

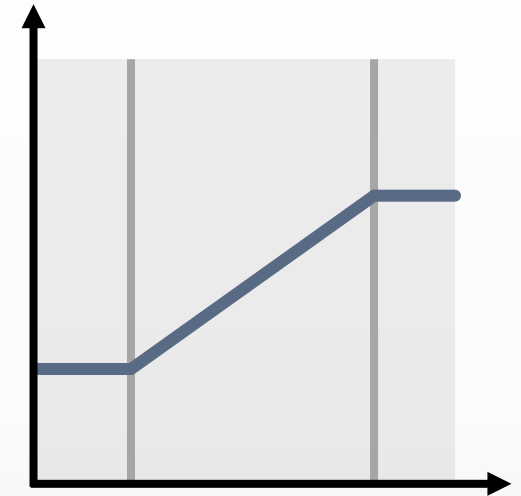
# Erhöhen der Last

Antwortzeit



Verbesserung!

Zeit



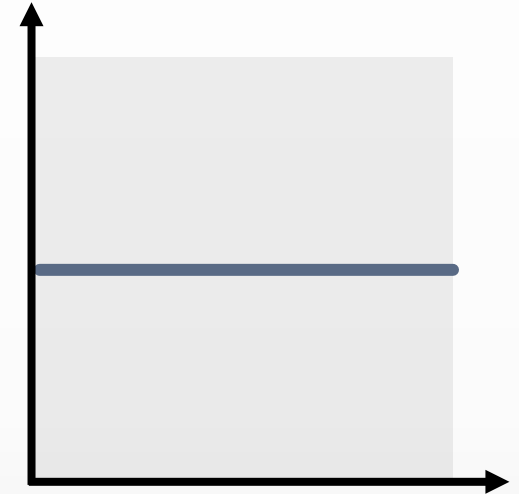
typisches  
Verhalten

# Konstante Last

Antwortzeit



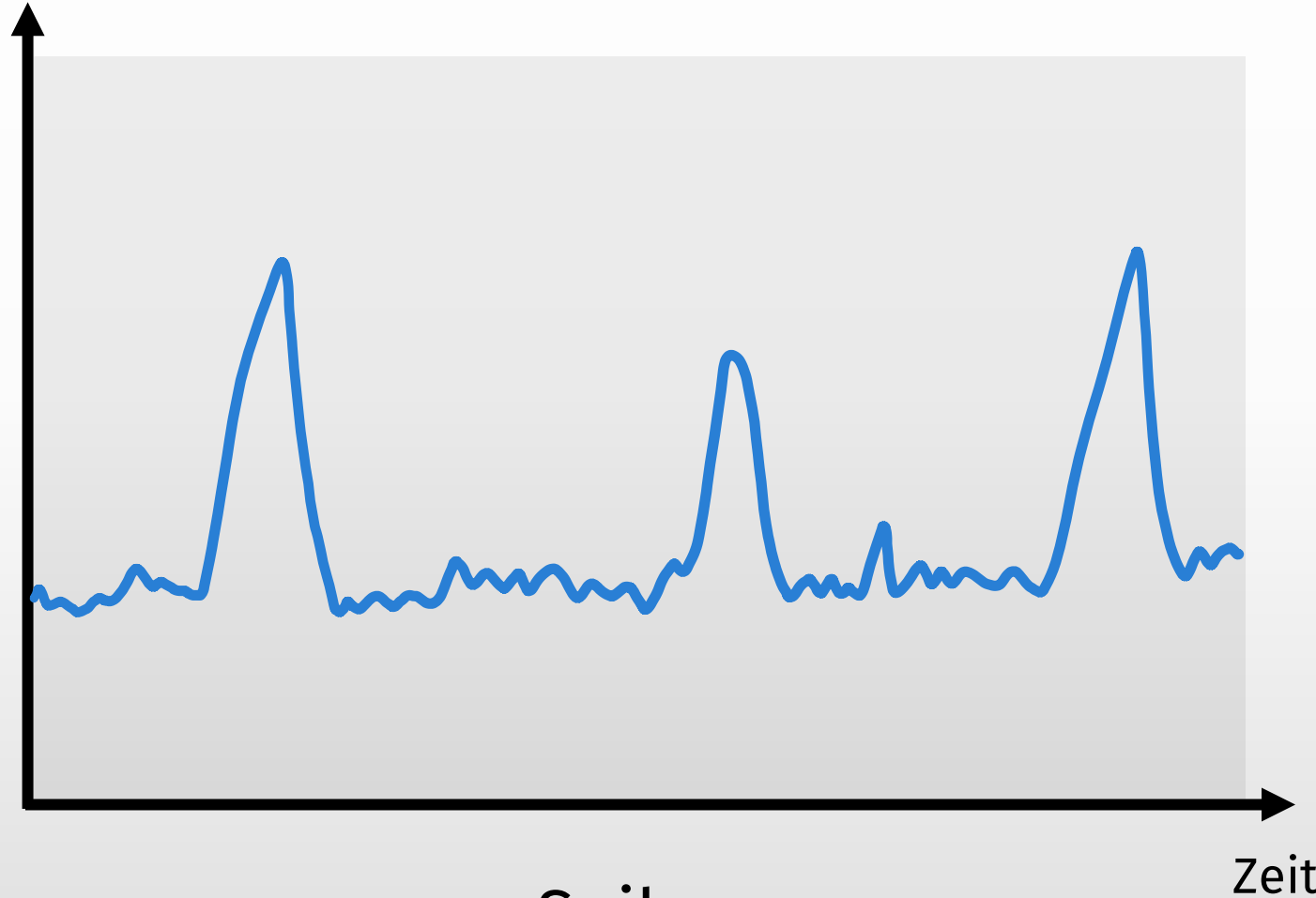
Langsame Verschlechterung



typisches  
Verhalten

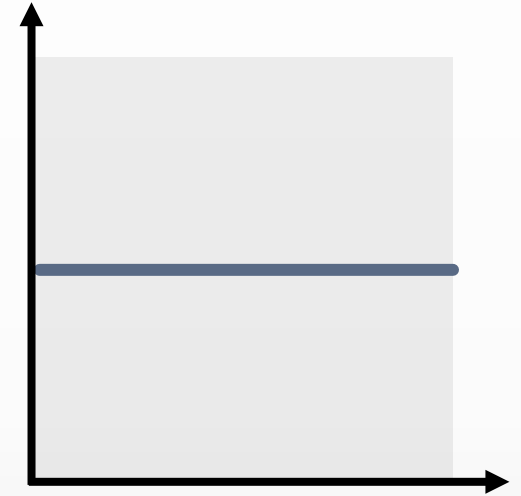
# Konstante Last

Antwortzeit



Spikes

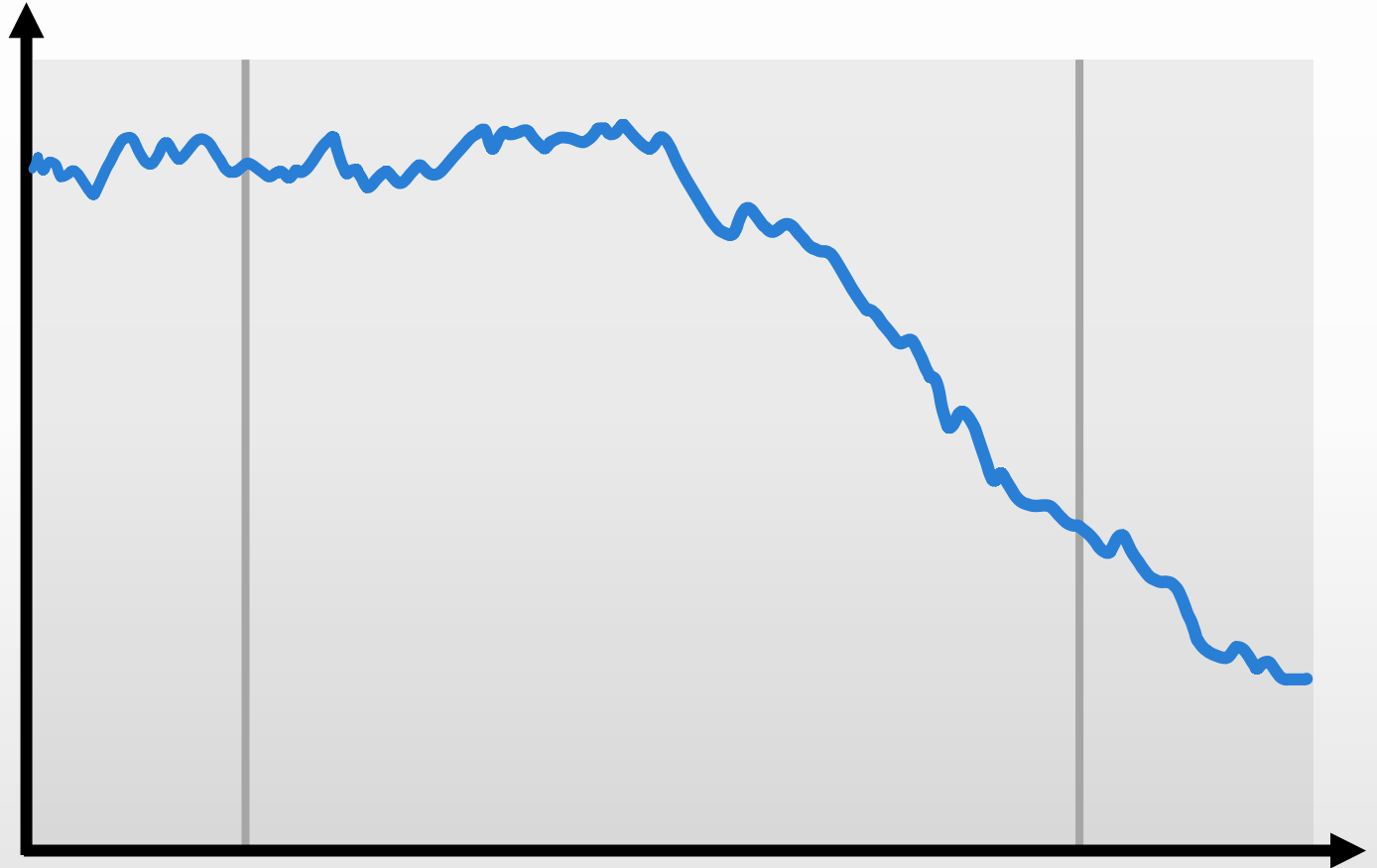
Zeit



typisches  
Verhalten

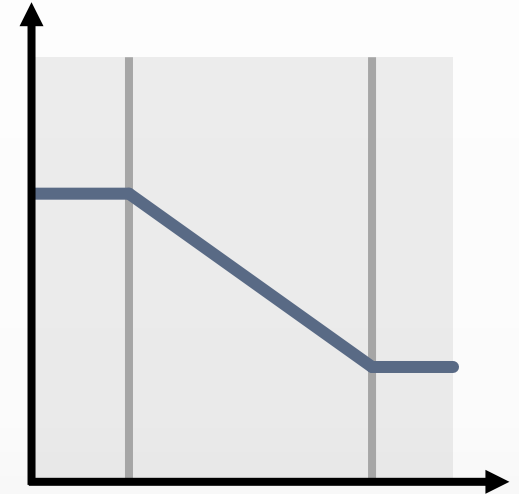
# Verringern der Last

Antwortzeit



«Verzögertes Ramp-Down»

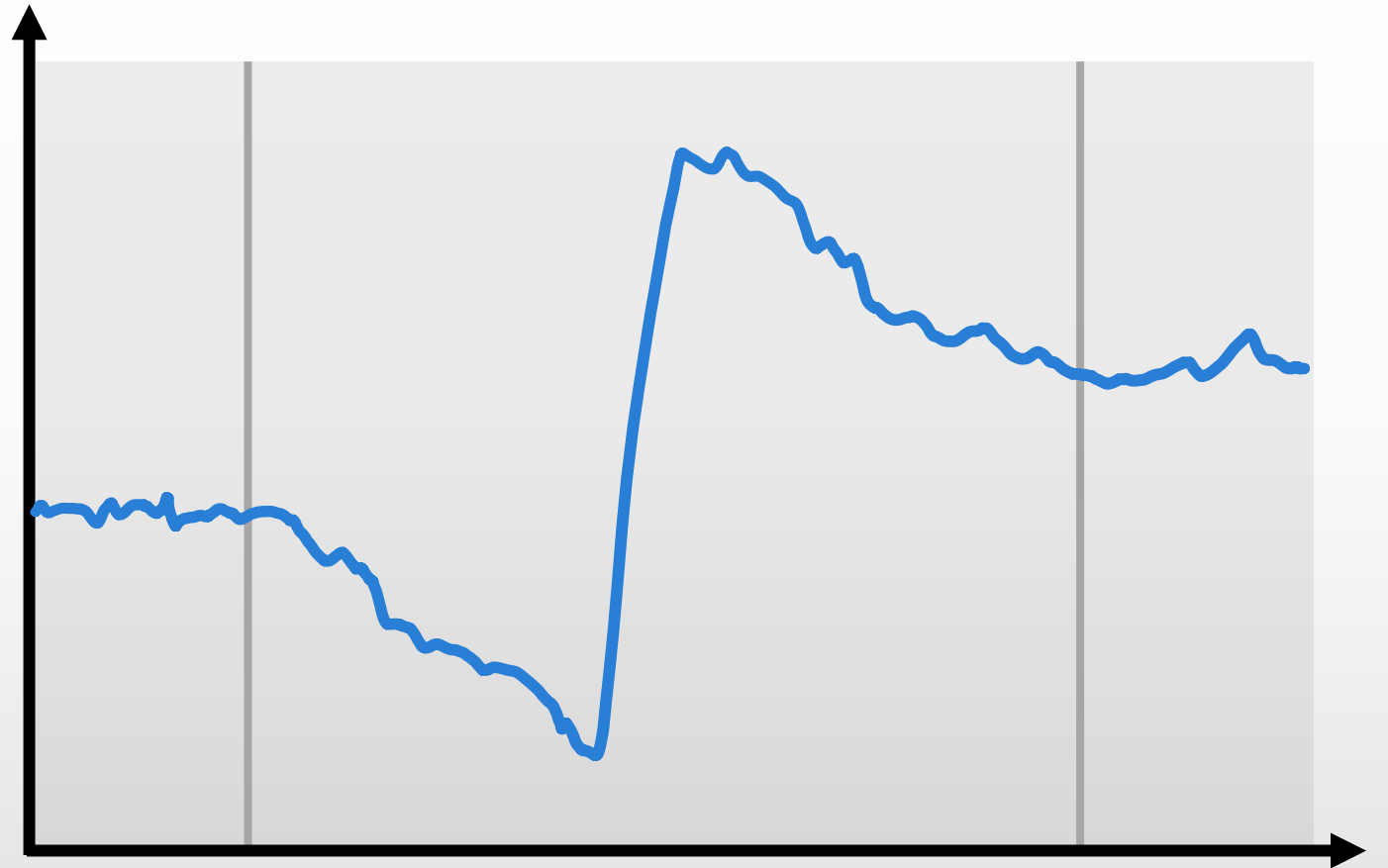
Zeit



typisches  
Verhalten

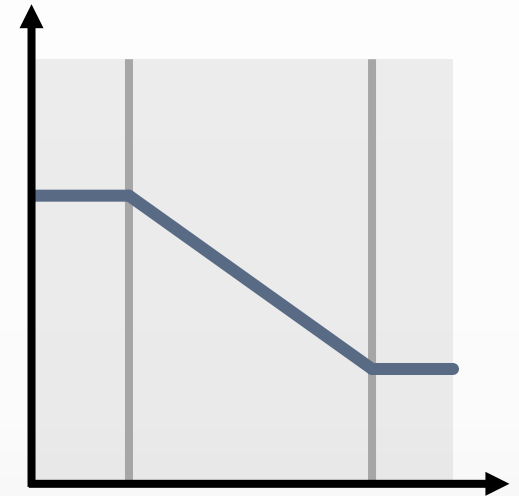
# Verringern der Last

Antwortzeit



Verschlechterung

Zeit



typisches  
Verhalten

**Fragen?**

