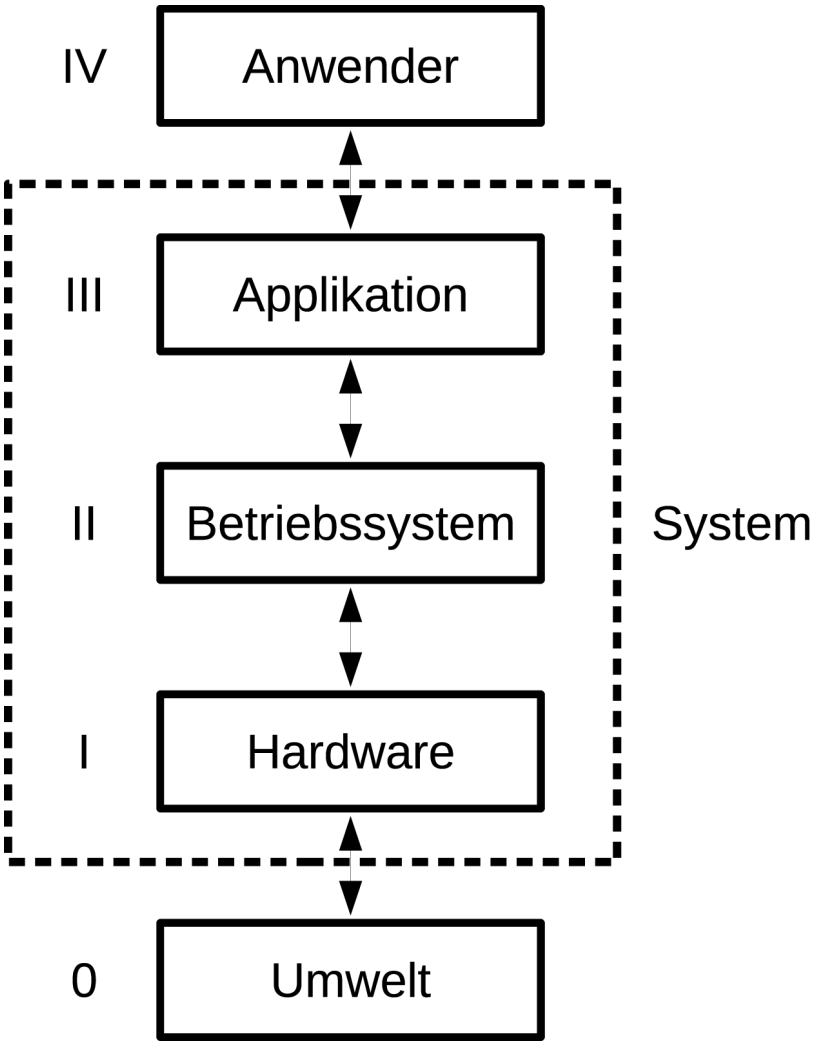
Daniel Koß

Vermeidung dynamischer Betriebsmittelverwaltung in sicherheitsgerichteten Echtzeitsystemen

Inhalt

- Einführung [~2 min]
- Stand der Technik [~10 min]
- Anforderungen der IEC 61508 [~5 min]
- Alternativen [~5 min]
- Fazit & Ausblick [~3 min]
- Fragen [∞]

Einführung

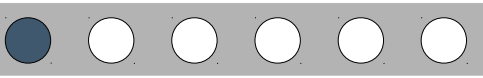

IV
Anwender
III
Applikation
II
Betriebssystem
System
I
Hardware
0
Umwelt

Einführung

- Ziele eines Betriebssystems
  - Abstraktion
  - Wiederverwendbarkeit
  - Verwaltbarkeit
  - Benutzbarkeit
  - Abschottung

Einführung

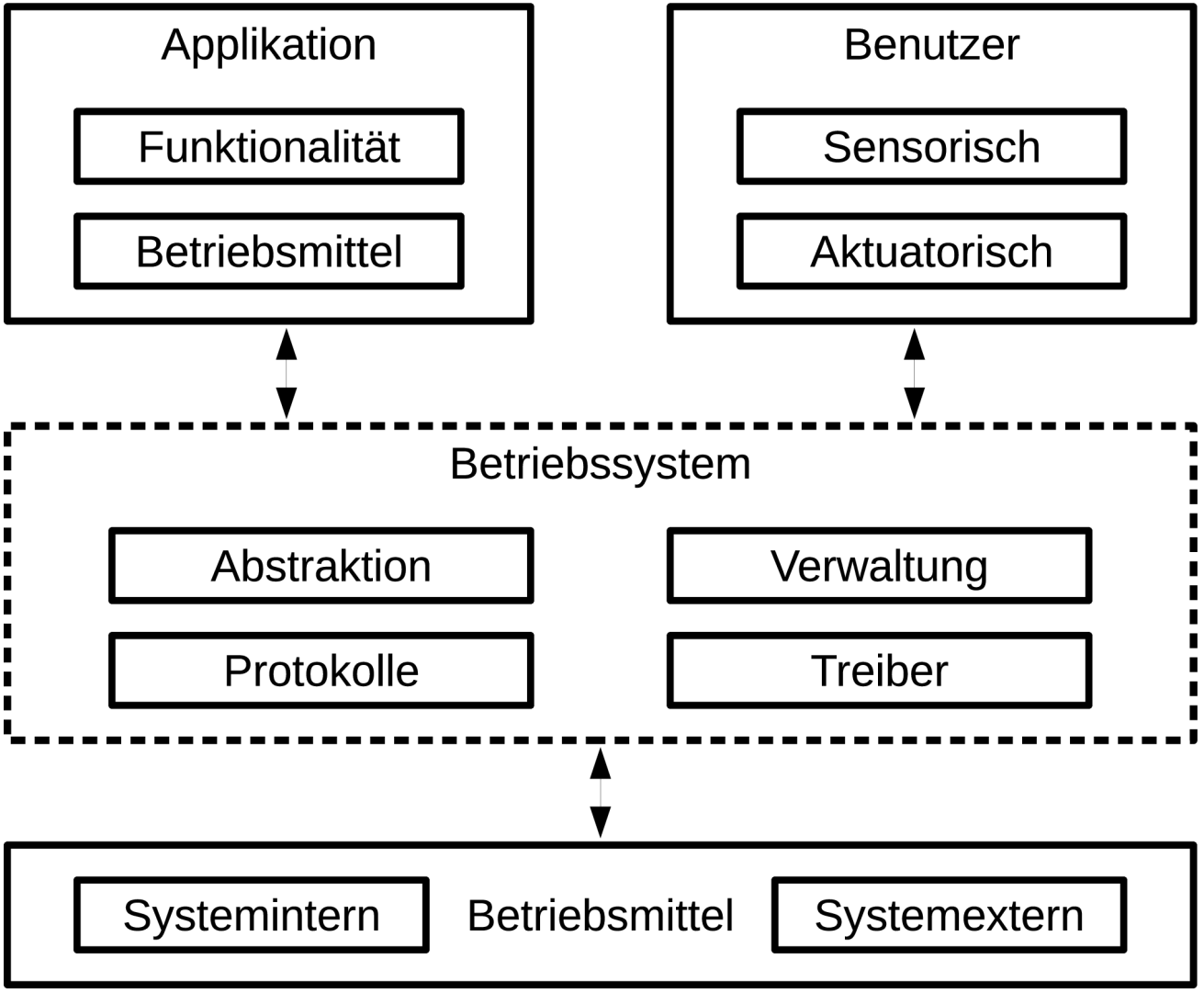
- Zusätzliche Ziele für Echtzeit-betriebssysteme
  - Einhaltung von Zeitbedingungen
  - Definiertes Ausfallverhalten
  - Definiertes Fehlverhalten



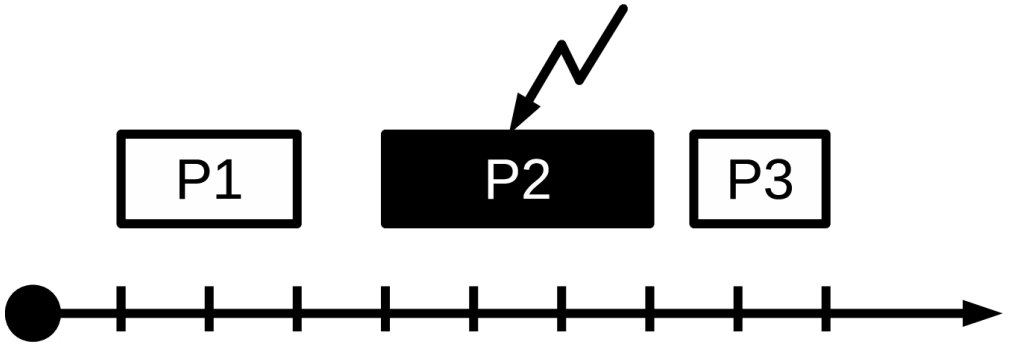
Einführung

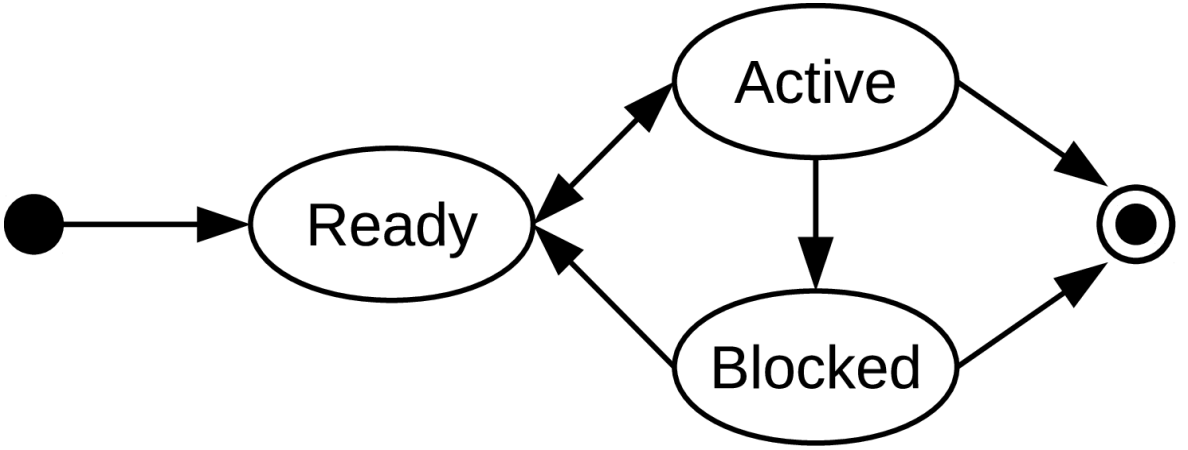
- Im Weiteren
  - Schnelle Initialisierung
  - Flexibilität, Skalierbarkeit
  - Zugänglichkeit
  - Benutzerfreundlichkeit

Stand der Technik: Komplexität

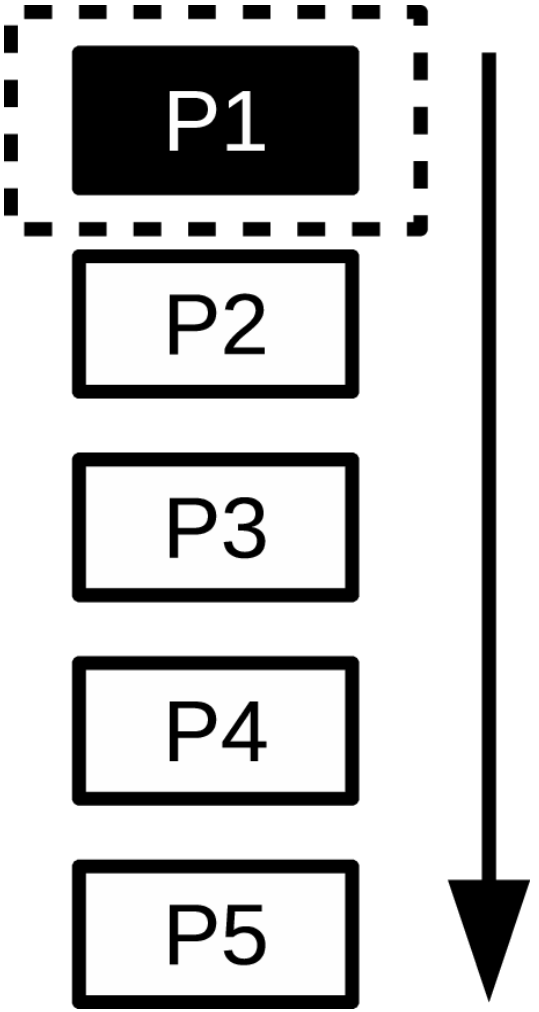


Stand der Technik: Nebenläufigkeit

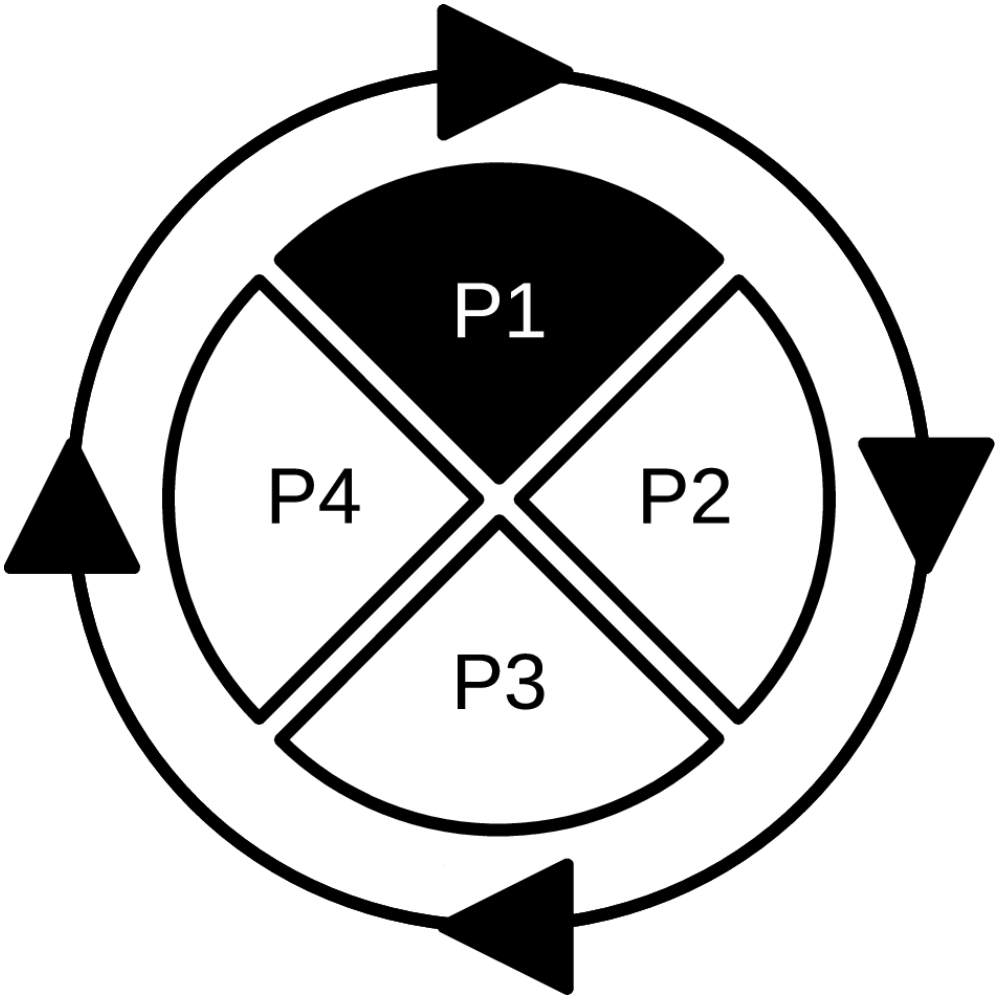




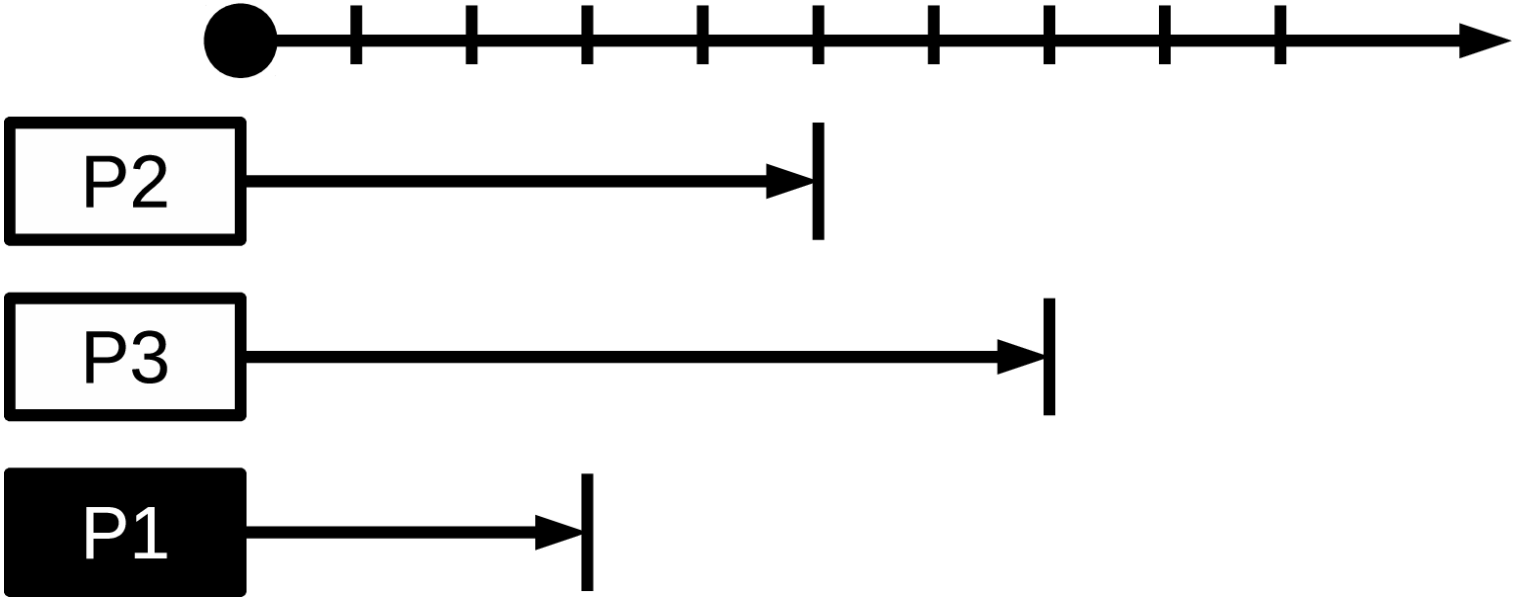
Stand der Technik: Prioritäten



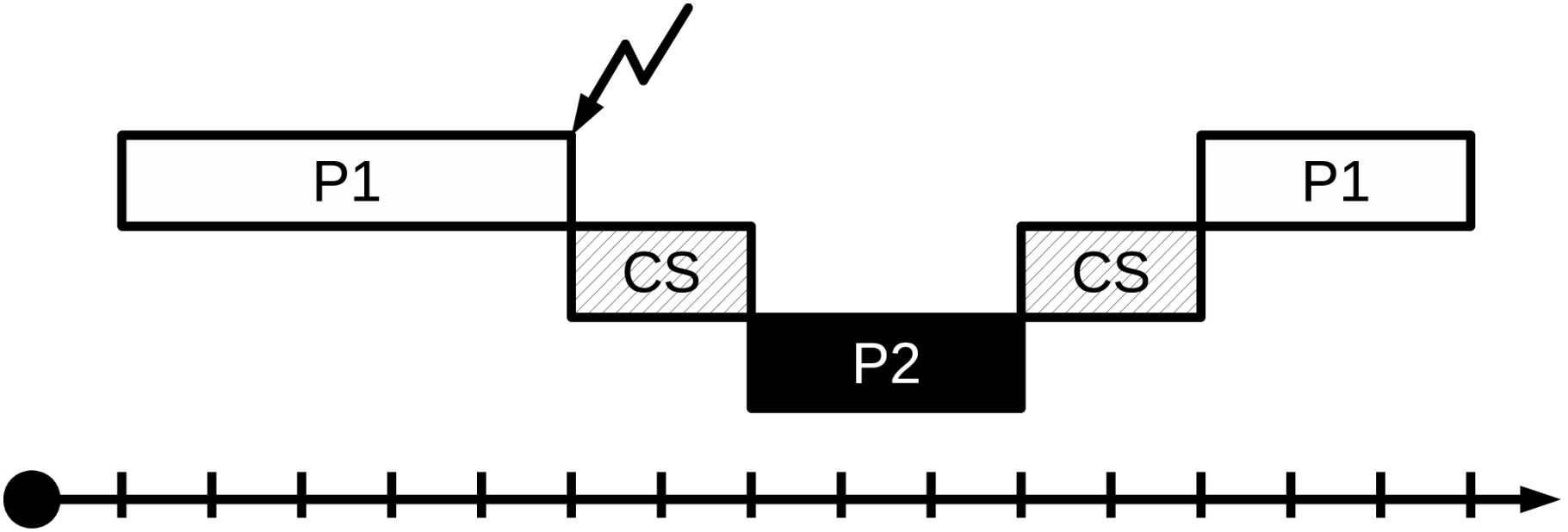
Stand der Technik: Zeitscheiben



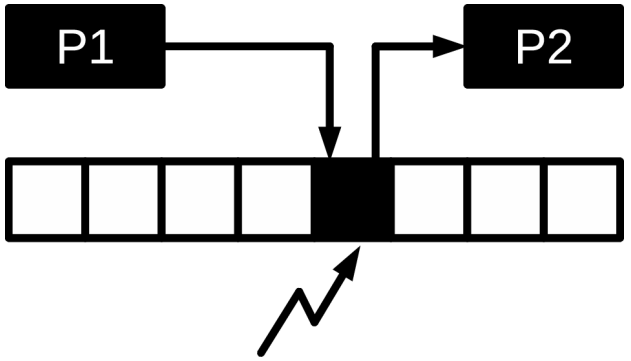
Stand der Technik: Zeitgrenzen

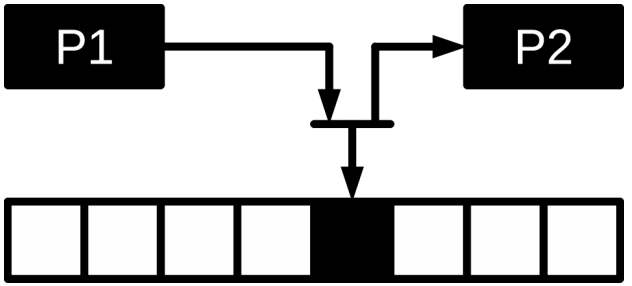


Stand der Technik: Unterbrechungen

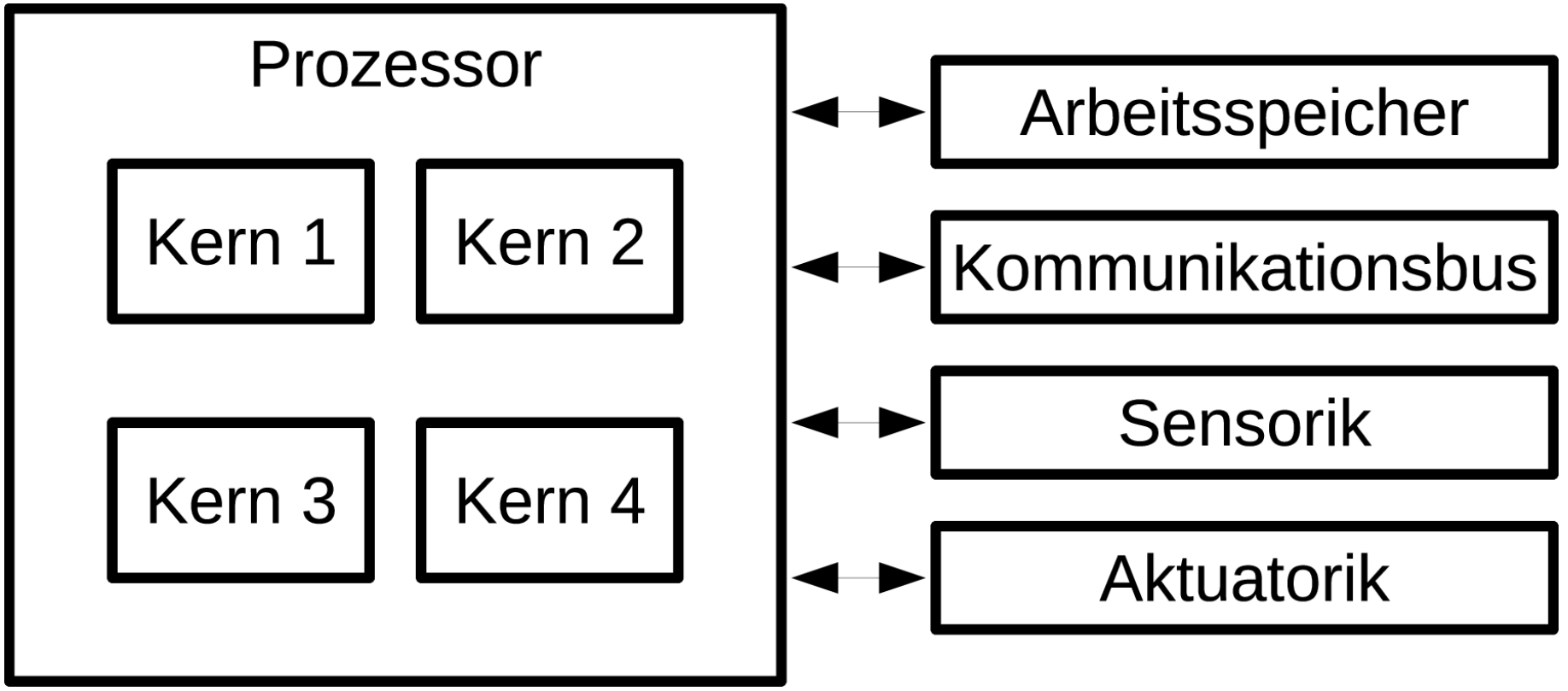


Stand der Technik: Interprozesskommunikation





Stand der Technik: Abstraktion



Anforderungen der IEC 61508

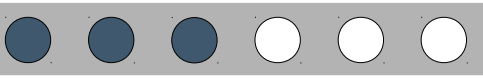
Was ist die DIN EN 61508?

- „Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme“
- 7-teilig
- Anforderungen an Software, Hardware, Management, Validierung
- Generische Norm zur funktionalen Sicherheit
- Abgeleitete Branchennormen

Anforderungen der IEC 61508

„Soweit es praktisch möglich ist, muss der Entwurf den sicherheitsbezogenen Teil der Software einfach halten.“

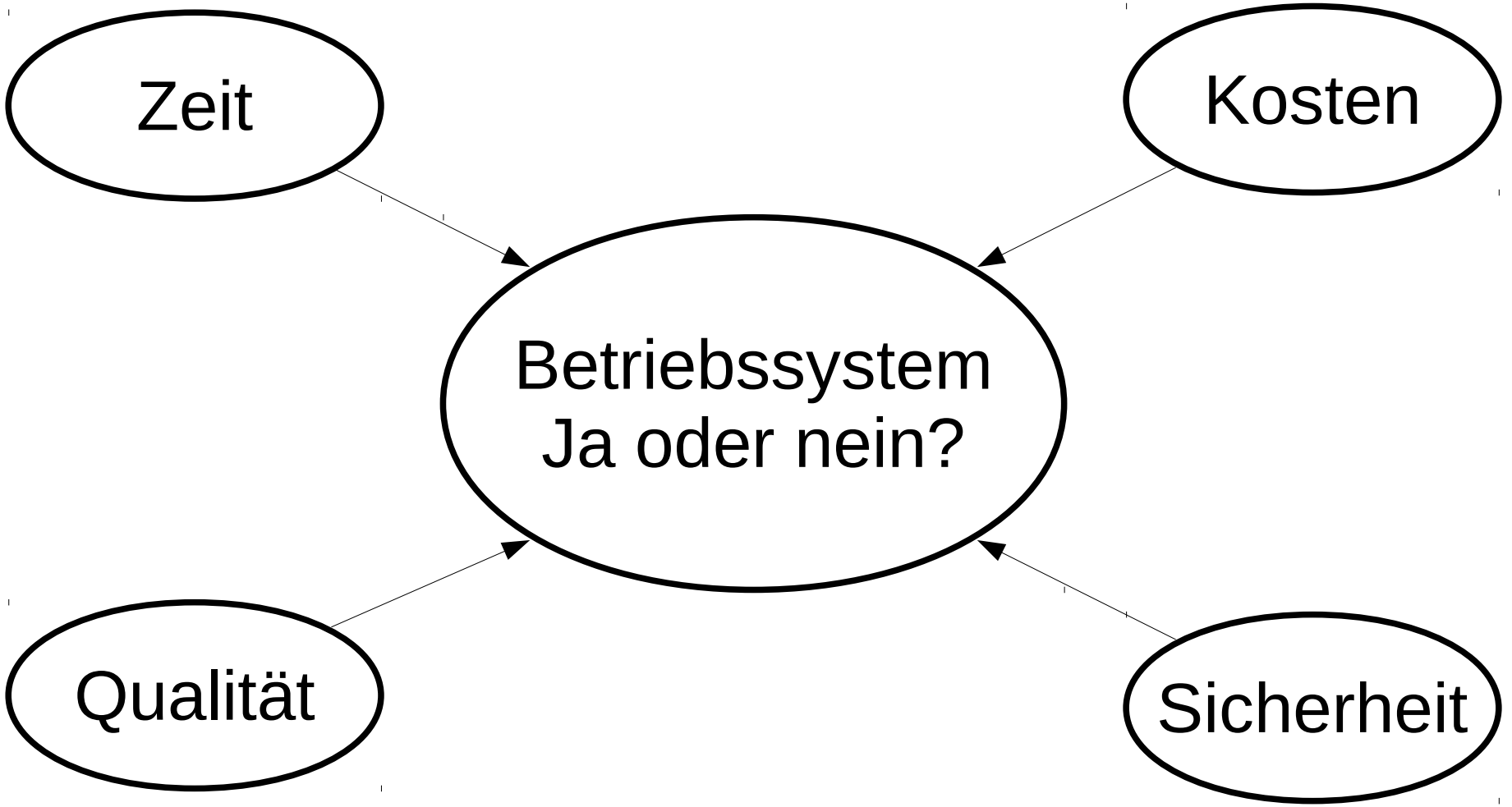
IEC 61508, Part 3: Software requirements



Anforderungen der IEC 61508

- Keine dynamische Betriebsmittelverwaltung
- Zustandsloses Softwaredesign
- Abgestufte Funktionseinschränkung
- Keine dynamische Rekonfiguration
- Zertifizierte/verifizierte Elemente
- Zyklische oder zeitgesteuerte Architektur
- Statische Synchronisation

Alternativen



Alternativen

Verhaltensabbildung direkt in Hardware

- Vorteile
  - Keine Zwischenstufen
    - → geringerer Fehlereinfluss durch Werkzeuge
  - Keine unnötige Hardware
- Nachteile
  - Schwere Verstehbarkeit
  - Schwere Partitionierung in Subsysteme/-funktionen → erhöhter interner Fehlereinfluss
  - Keine Betriebsbewährtheit

Alternativen

Hardwarenahe Software

- Vorteile
  - Direkter Zugriff auf Maschinenfunktionen
  - Minimalität der Software
- Nachteile
  - Schlechte Portierbarkeit/Wiederverwendbarkeit
  - Schwere Verstehbarkeit

Alternativen

Rahmenwerke

- Vorteile
  - Abstraktion
  - Verifiziert und evtl. betriebsbewährt
- Nachteile
  - Teilweise proprietär → schwer einsehbar/verstehbar
  - Umfangreicher als nötig → Fehlereinflüsse

Alternativen

Verbundene Werkzeugketten

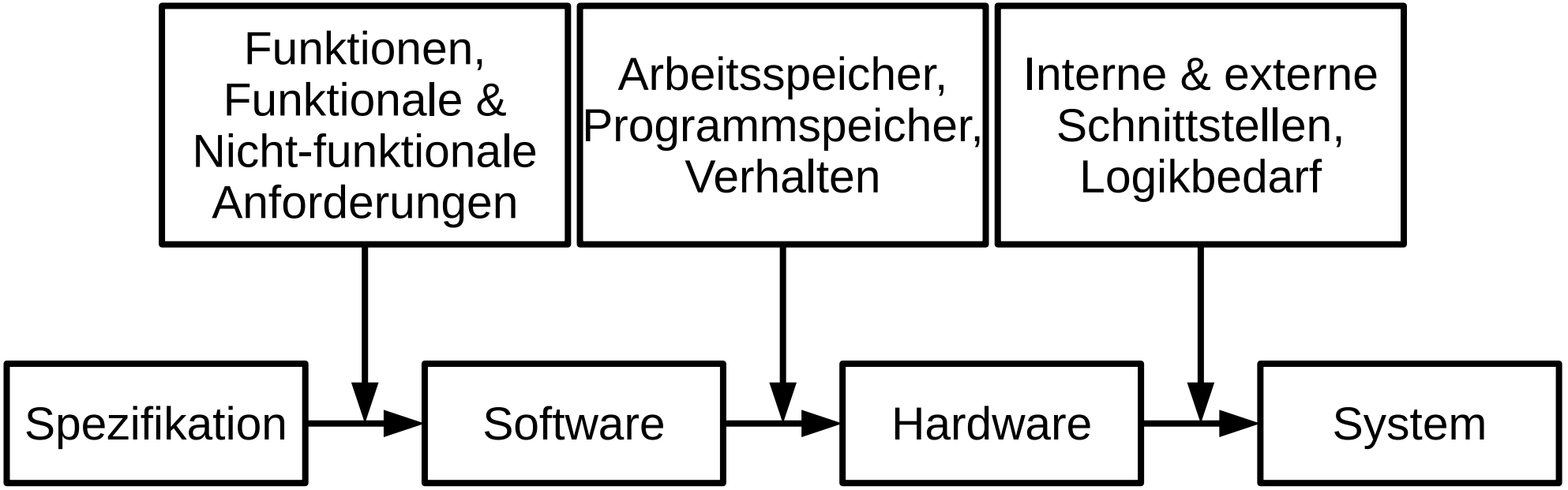
- Vorteile
  - Teilautomatisierbar
  - Wiederverwendbarkeit von Teilergebnissen
- Nachteile
  - Komplexe Interaktionen → schwer nachvollziehbar
  - Komplexe Werkzeuge → Fehlereinflüsse
  - Keine Betriebsbewährtheit

Fazit

- Betriebssysteme sind nicht geeignet für sicherheitsrelevante Umgebungen
  - Hohe Komplexität
  - Schlechte Testbarkeit von Zeit- & Ausfallverhalten
  - Schlechte Möglichkeiten der Fehlersuche
  - Keine Garantie für die Einhaltung von Zeitgrenzen
  - Kein deterministisches Verhalten
  - Inhärentes Problem der knappen Betriebsmittel nicht gelöst
- sicherheitsrelevanter Teil der Software muss so einfach wie möglich gehalten sein (IEC 61508)

Ausblick

- Parametrierbare Entwurfsphasen mit betriebsmitteladäquatem Resultat



Ende

Vielen Dank
für die Aufmerksamkeit!