



Web- und App-Programmierung  
**Deployment und  
Software Reviews**

**Prof. Dr.-Ing. Tenshi Hara**  
tenshi.hara@ba-sachsen.de

# AUFBAU DER LEHRVERANSTALTUNG



# Deployment

## SPF PATTERN / MONOLITH (1/2)

- monolithische Anwendung (Single Point of Failure)
- i.d.R. Top-Down oder Bottom-Up implementiert (ggf. mit Black-Box Modulen)
- schnell erzeugt, skaliert gut für kleine Anwendungen
- ungünstig bei multiplen/spezialisierten Adaptionstrategien
- 2 Hauptvarianten:
  - Client-Server-Koexistenz
  - Client-Server-Symbiose

## SPF PATTERN / MONOLITH (2/2)

- Client-Server-Koexistenz
  - häufig
  - ohne Optimierungen
  - Server kann unterschiedliche Clients versorgen
  - Client kann unterschiedliche Server verwenden
- Client-Server-Symbiose
  - eher selten
  - hochgradig optimierte Implementierung
  - Server und Client arbeiten integriert (voneinander abhängig)

## MICRO SERVICES (1/2)

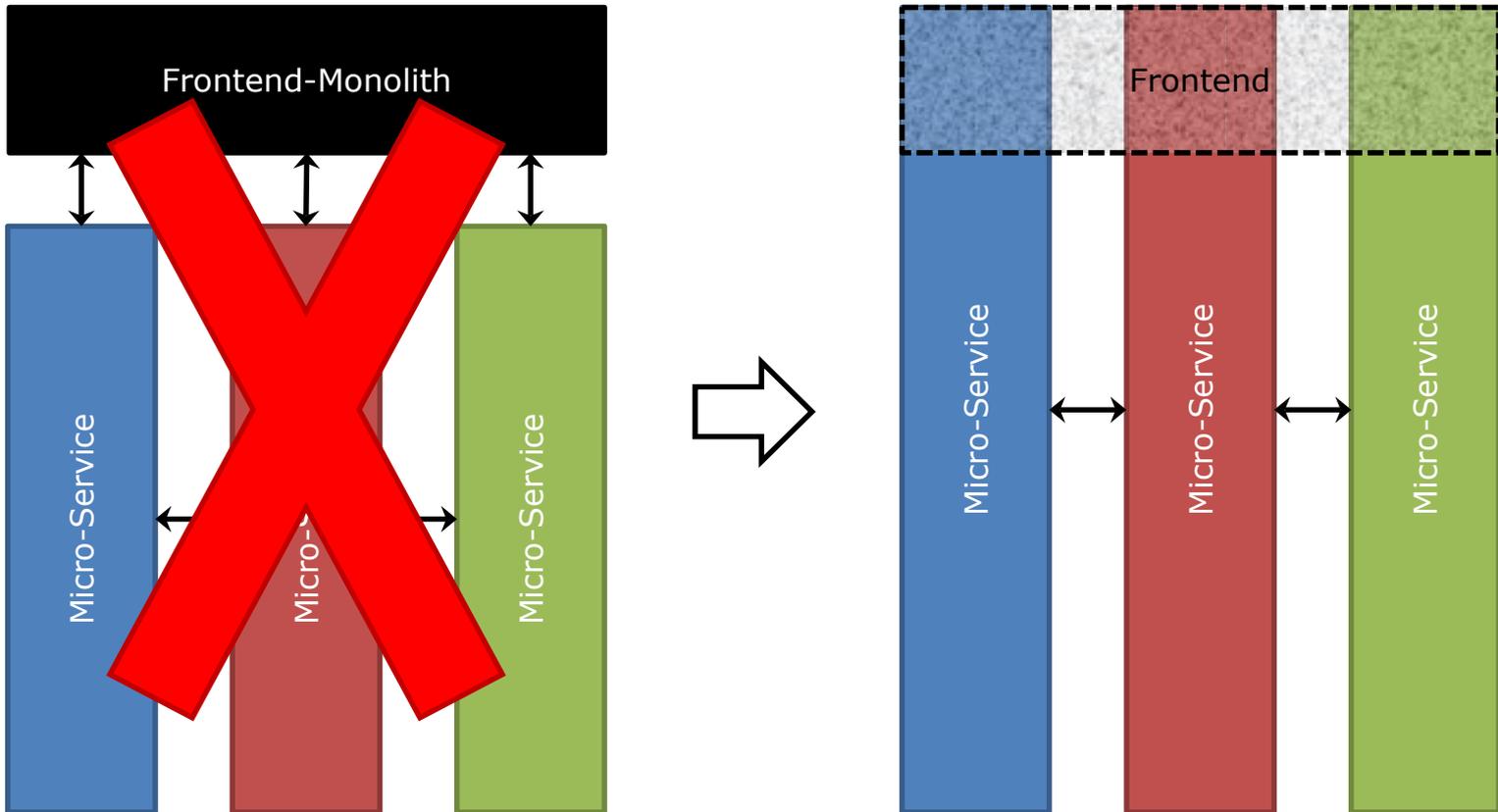
Grundidee: für jede Aufgabe gibt es einen eigenen replizierbaren Service

- alle Komponenten erledigen exakt eine Aufgabe  
(entsprechend Unix-Philosophie: Do one Thing and do it well)  
→ Kontextbindung im Sinne des Domain-driven Designs
- alle Komponenten sind austauschbar
  - arbeiten gegen definierte Schnittstellen → Obfuskation von Interna
  - sind auffindbar und sofort nutzbar → Dienstkomposition
- multiple entkoppelte Instanzen eines Services
  - arbeiten parallel → Load Balancing
  - behindern sich nicht → Isolation

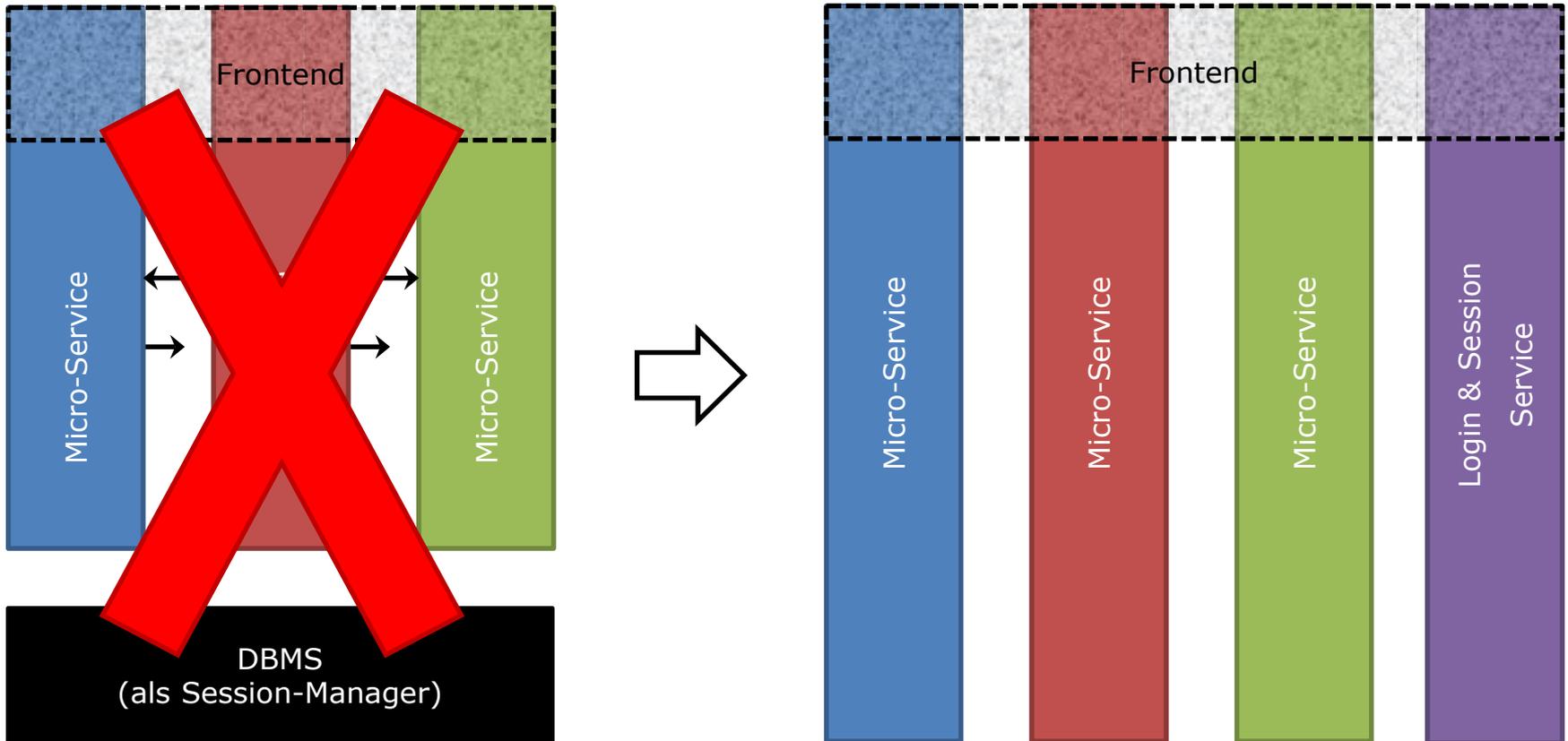
## MICRO SERVICES (2/2)

- benötigen i.d.R. Zeitsynchronisierung → Lamport-Ansatz
- Möglichkeit des Distributed Testing Desasters
- Probleme im Sinne des CAP-Theorem; i.d.R. behandelt durch
  - Aufweichung der Konsistenz
  - NoSQL-DBMS und BASE-Prinzip  
(Basically Available, Soft State, Eventual Consistency)
- (sehr) anfällig für Entwurfsfehler der Entwickler → Frontend Monolith

# MICRO SERVICES ANTI-PATTERN (1/2)



# MICRO SERVICES ANTI-PATTERN (2/2)



## MICRO SERVICE FRONTEND INTEGRATION (1/6)

- einheitliches Look & Feel
  - Grundvoraussetzung für eine hohe Usability
  - Gestaltungsdisziplin, ggf. Corporate Design notwendig
  - rein optisch muss alles aus einem Guss zusammengehörig wirken
- unabhängige Deploybarkeit
  - Benutzeroberflächen der einzelnen Micro Services müssen unabhängig voneinander deploybar sein
  - im Vorfeld weder Quellcode noch Softwareartefakte zwischen Services austauschen
    - dynamische Integration der GUI zur Laufzeit
- Zustandsbasierte Kommunikation
  - Schnittstellen definieren
  - Zustandsinformationen (und ggf. Nachrichten) definieren

## MICRO SERVICE FRONTEND INTEGRATION (2/6)

- Technologieunabhängigkeit
  - jede einzelne GUI sollte sich technologieunabhängig umsetzen lassen
  - Unterstützung verschiedener Frameworks (Angular, React, ...) innerhalb eines Frontends sinnvoll
- Isolation
  - GUIs immer voneinander isolieren!
  - Drittanbieter-Service (Like-Button, Werbung, ...) in Sandbox einbetten

### Wichtigstes Integrationswerkzeug bei Web-Anwendungen: Shell Loading

⇒ egal welcher Service zuerst angesprochen wird:

alle anderen GUI-Komponenten per iframe nachladen und integrieren

⇒ Kommunikation zwischen den Frames mittels JavaScript

## MICRO SERVICE FRONTEND INTEGRATION (3/6)

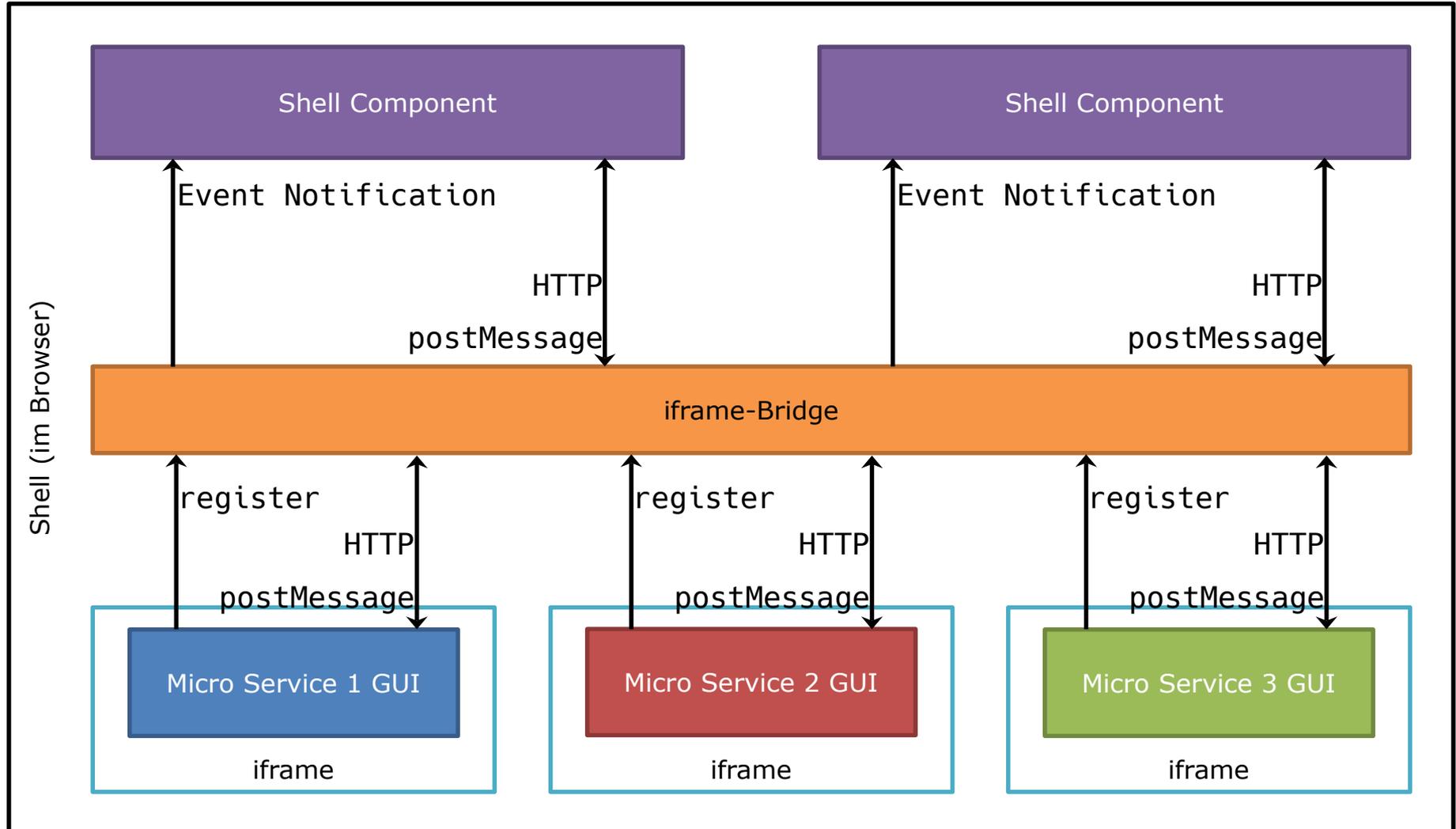
### JavaScript-Kommunikation zwischen iframes

```
// Sender
const iframe =
(<HTMLIFrameElement>document.getElementById("iframeName"));
iframe.contentWindow.postMessage(
  {"message": "Hallo Welt!"},
  "https://example.com/"
);

// Empfänger
export class Component implements EventListenerObject {
  constructor()
    window.addEventListener('message', this);
  public handleEvent(event)
    if (isTrustedURL(event.origin) )
      console.log(message received:' + event.data.message);
}
```

# MICRO SERVICE FRONTEND INTEGRATION (4/6)

## Zusammenführen mittels iframe-Bridge



## MICRO SERVICE FRONTEND INTEGRATION (5/6)

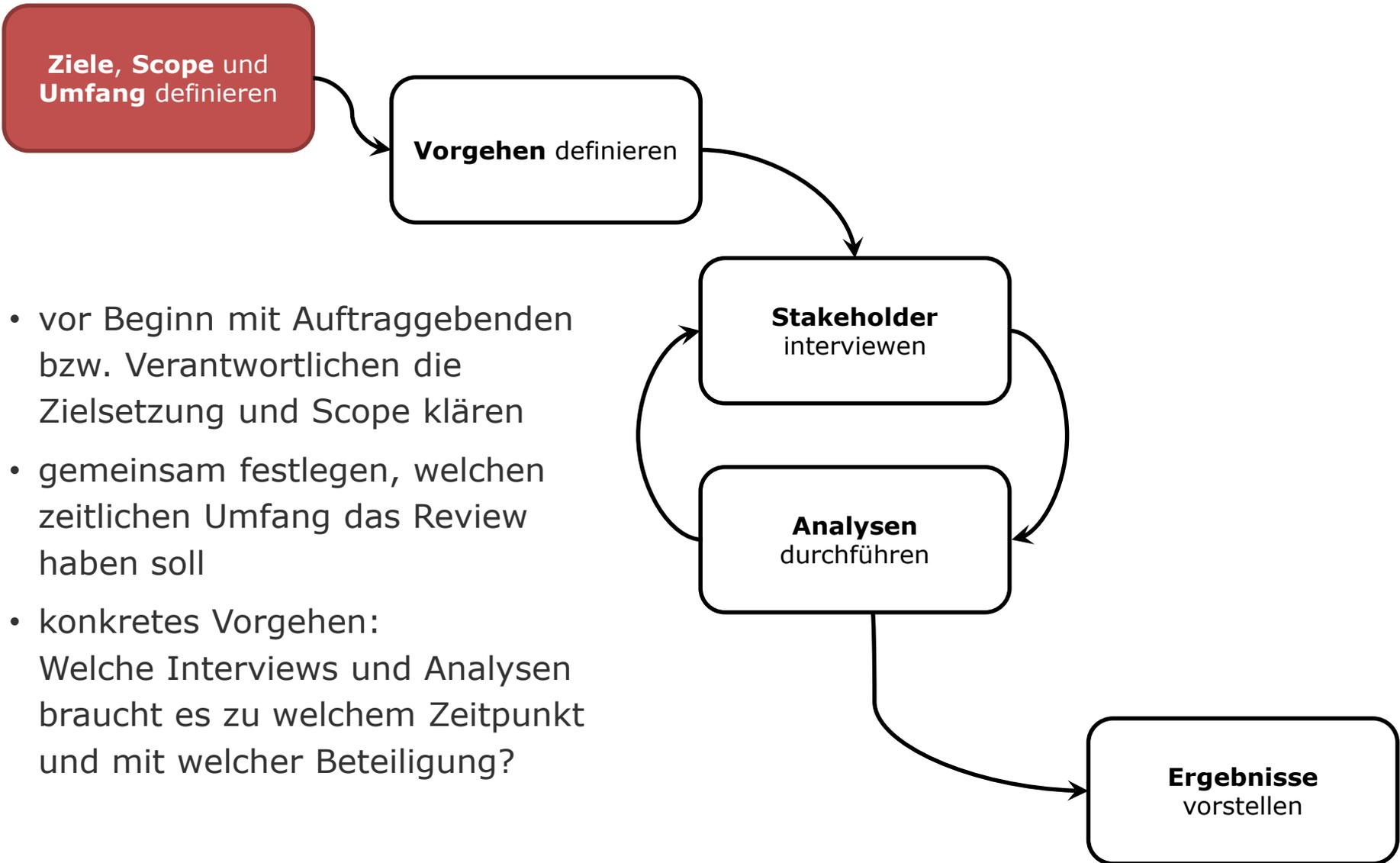
```
export class IFrameBridge {
  private iFrameMap = new Map<string, IFrame>();
  public registerIFrame(event) {
    const url = event.target.src;
    if (isTrustedURL(url))
      this.iFrameMap.set(
        url, new IFrame(url, event.target.id)
      );
  }
  public postMessageToIFrame(url: string, message: any) {
    if (this.iFrameMap.has(url))
      this.iFrameMap.get(url).postMessage(message);
  }
  // ...
}
```

## MICRO SERVICE FRONTEND INTEGRATION (6/6)

```
class IFrame {
    private id: string;
    private url: string;
    private htmlIFrameWindow: Window;
    constructor(url: string, id: string) {
        this.url = url;
        this.id = id;
        this.htmlIFrameWindow = (
            <HTMLIFrameElement>document.getElementById(this.id)
        ).contentWindow;
    }
    public postMessage(message)
        this.htmlIFrameWindow.postMessage(message, this.url);
    // ...
}
```

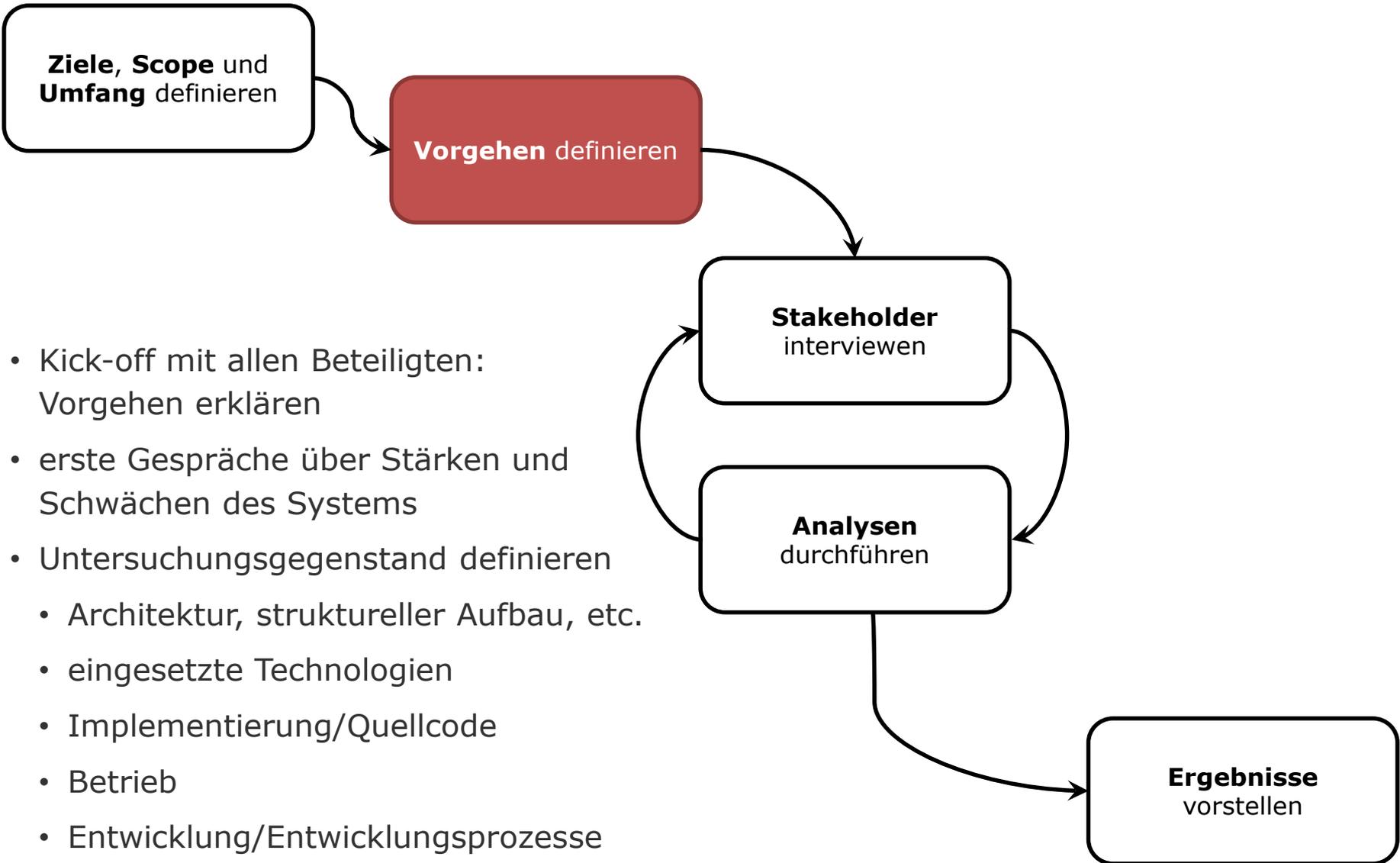
# Software Review

# 5 PHASEN DES REVIEWS



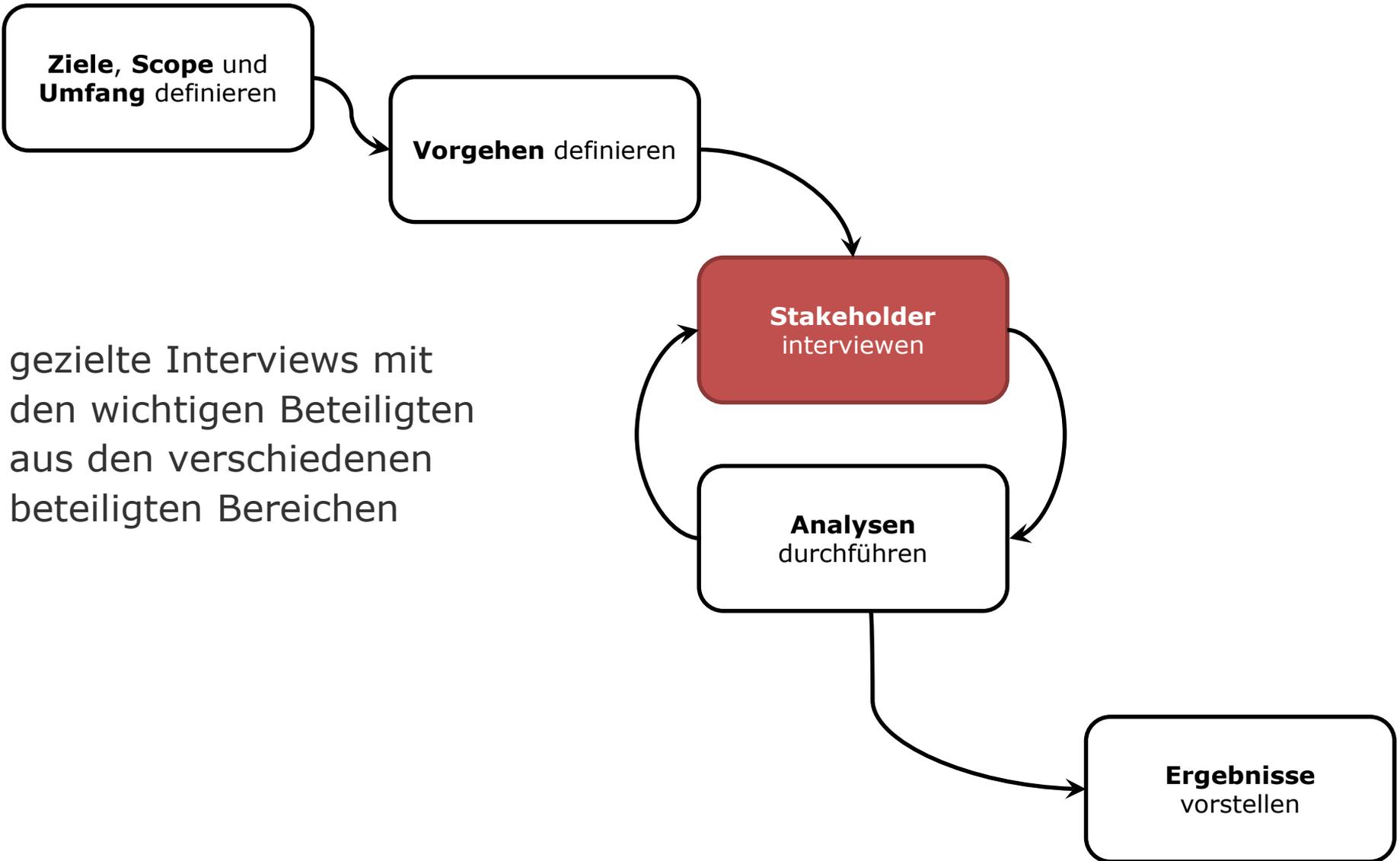
- vor Beginn mit Auftraggebenden bzw. Verantwortlichen die Zielsetzung und Scope klären
- gemeinsam festlegen, welchen zeitlichen Umfang das Review haben soll
- konkretes Vorgehen: Welche Interviews und Analysen braucht es zu welchem Zeitpunkt und mit welcher Beteiligung?

# 5 PHASEN DES REVIEWS

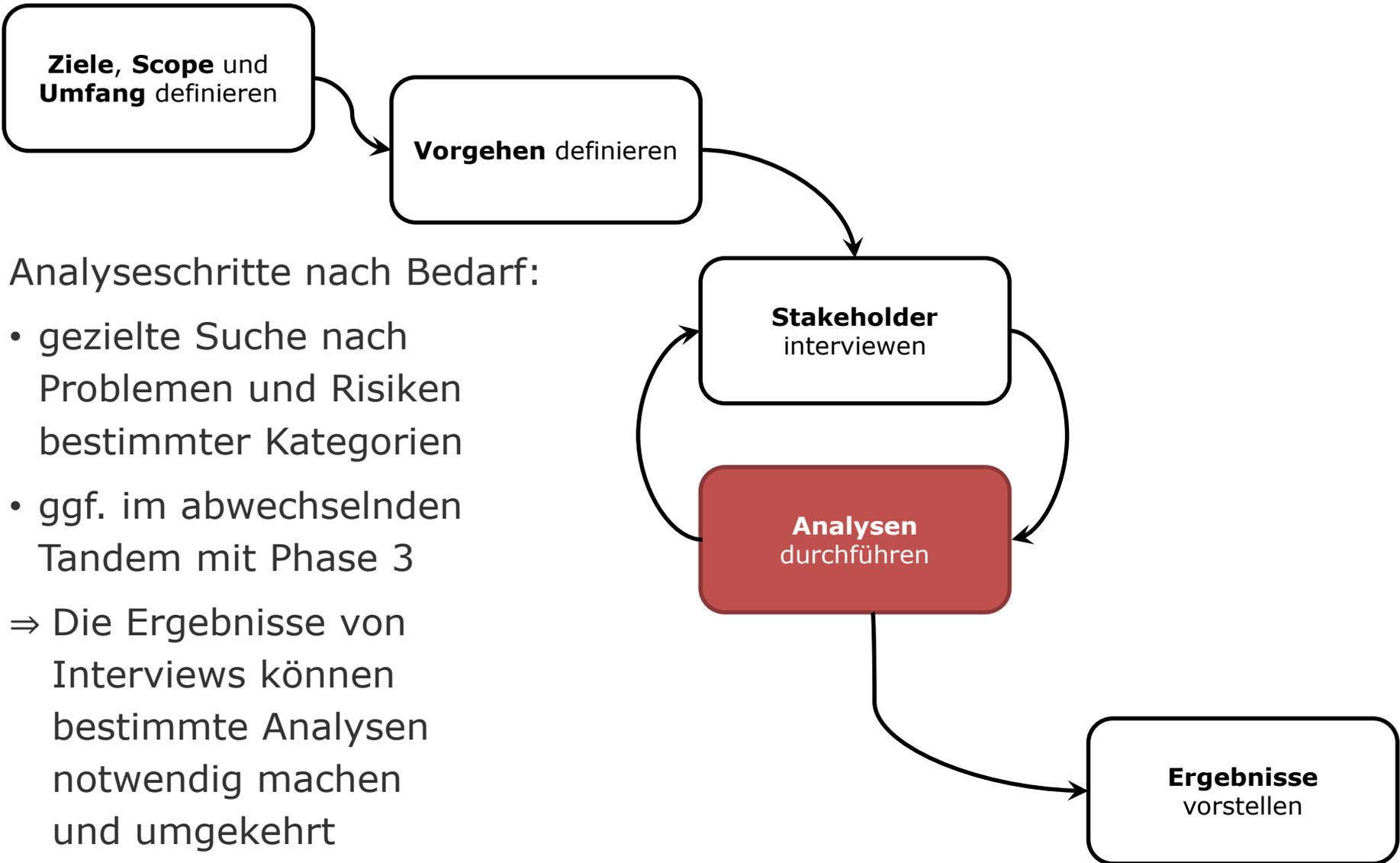


- Kick-off mit allen Beteiligten: Vorgehen erklären
- erste Gespräche über Stärken und Schwächen des Systems
- Untersuchungsgegenstand definieren
  - Architektur, struktureller Aufbau, etc.
  - eingesetzte Technologien
  - Implementierung/Quellcode
  - Betrieb
  - Entwicklung/Entwicklungsprozesse

# 5 PHASEN DES REVIEWS



## 5 PHASEN DES REVIEWS

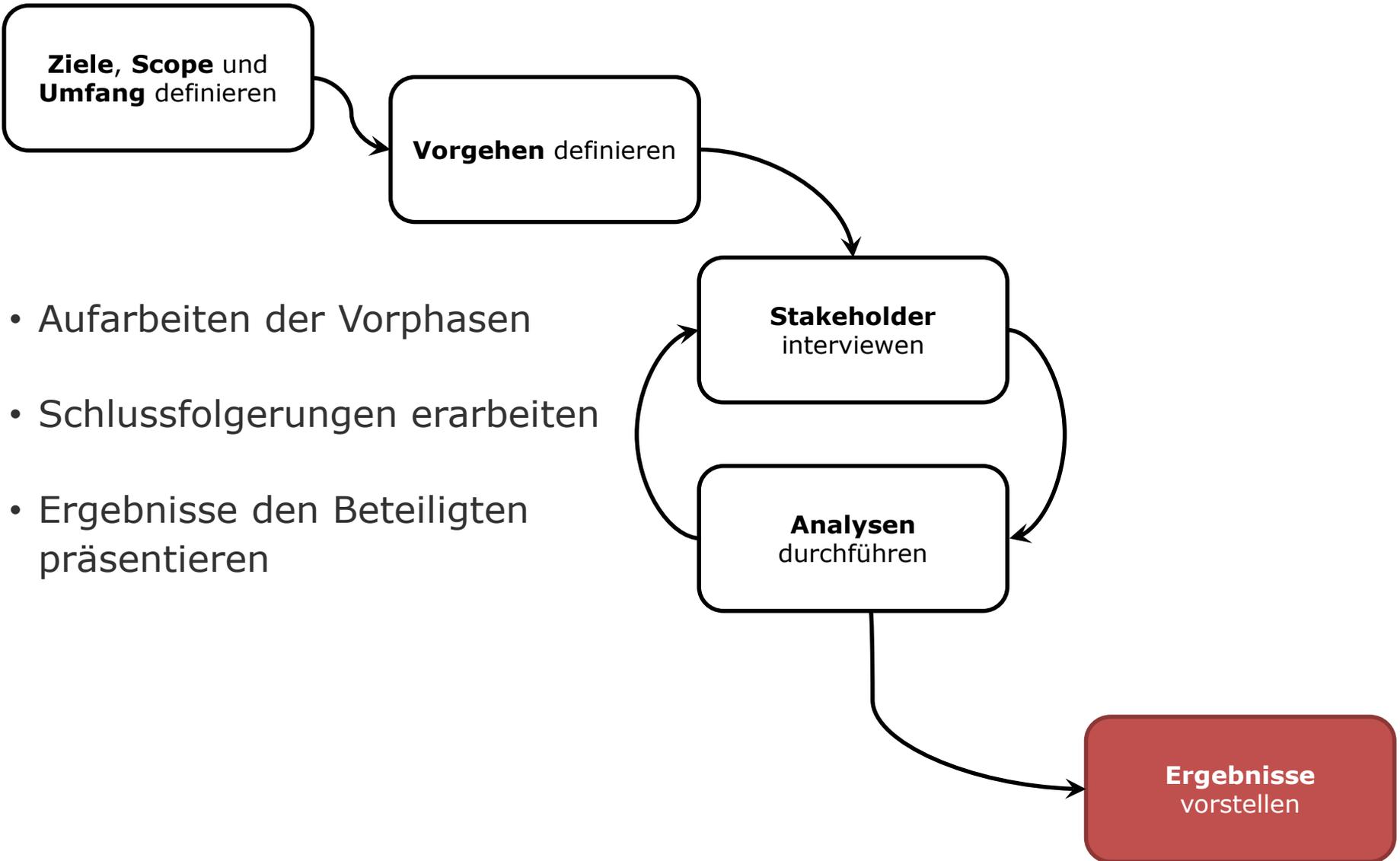


Analyseschritte nach Bedarf:

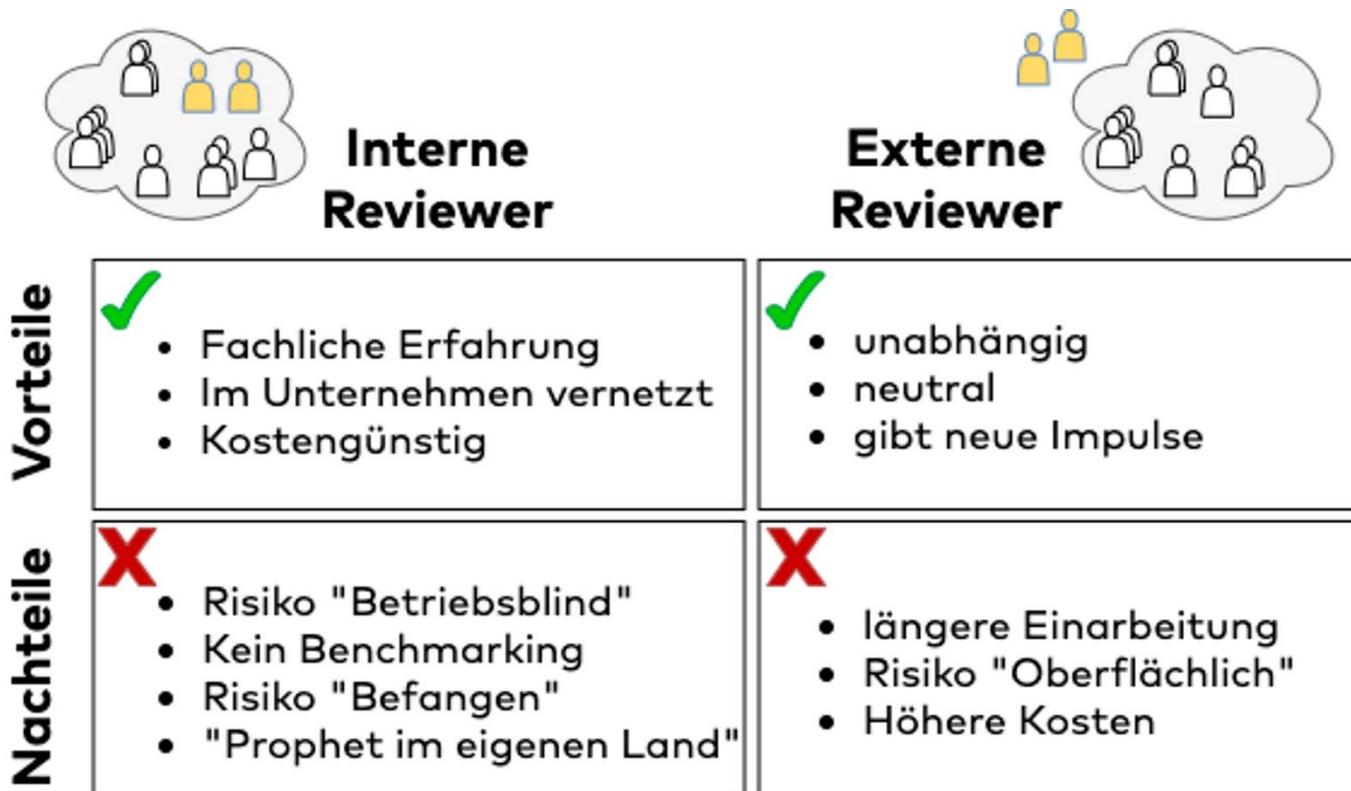
- gezielte Suche nach Problemen und Risiken bestimmter Kategorien
- ggf. im abwechselnden Tandem mit Phase 3

⇒ Die Ergebnisse von Interviews können bestimmte Analysen notwendig machen und umgekehrt

# 5 PHASEN DES REVIEWS



# WER REVIEWT?



© 2020 heise online

# UNTERSUCHUNGSGEGENSTÄNDE

Bereich	Was zu untersuchen ist
Architektur	Komponenten, Subsysteme, Schnittstellen, Abhängigkeiten, Kopplung, Kohäsion, Konsistenz
Code	Strukturierung, Komplexität, Änderungshäufigkeit, Verständlichkeit, Kopplung, Einheitlichkeit
Technologie	eingesetzte Basistechnologie, 3rd-Party
Qualität	Erreichung von Qualitätsanforderungen (z.B. Performance, Änderbarkeit, Robustheit, Benutzbarkeit, Betreibbarkeit)
Kontext	externe Schnittstellen, externe Datenquellen und -senken, Benutzerrollen
Infrastruktur	verwendete Infrastruktur, Prozessoren, Netzwerke, Speichermedien, etc.
Daten	Datenstrukturen, DB/DBMS, Korrektheit und Volatilität, Replikation, Backup
Laufzeitverhalten	Laufzeitverhalten und Speicherverhalten, allgemeine Ressourcennutzung, Bottlenecks
Entwicklungsprozess	Requirements- und Change-Management, Entwicklung, Test, Build, Deployment, Versionsmanagement
Dokumentation	Dokumentationsumfang, Aktualität/Korrektheit und Akzeptanz, Konsistenz
Management	Umgang mit Zeit und Budget, Ressourcenplanung, Organisation

# INTERVIEWPARTNER

- Nutzer
- fachlich Verantwortliche
- technisch Verantwortliche
- Auftraggeber, Management
- Entwicklungsteam bzw. Mischung aus mehreren Teams
- Test/Qualitätssicherung  
(falls nicht Teil des Entwicklungsteams)
- Personen aus Infrastruktur und/oder Betrieb
- Projekt- und Produkt-Manager
- in agilen Organisationen:  
Product Owner, Scrum Master oder Agile Coach

# ABSCHLUSSPRÄSENTATION

1. kurze, prägnante Management-Summary
2. Zusammenfassung organisatorischer Aspekte des Reviews
  - Was genau waren die Ziele und Scope des Reviews?
  - Welche Beschränkungen gibt es?
  - Wie wurde vorgegangen? Wie viel Zeit wurde (circa) worin investiert?
  - Mit wem hat wer worüber gesprochen, inklusive Daten und Dauer?
  - Welche Aktivitäten waren neben Gesprächen/Interviews Bestandteil des Reviews?
3. Welche Aspekte am System und dessen Entwicklung waren positiv zu bewerten?
4. Zusammenfassung der Probleme und Risiken
  - In welchen Kategorien waren Probleme und Risiken zu finden?
  - Welche Probleme gab es, beginnend mit den höchsten Prioritäten?
  - Wo drohen Risiken?
  - Welche Auswirkungen sind zu erwarten oder sind bereits akut?
5. Zusammenfassung der vorgeschlagenen Maßnahmen
  - Maßnahmen in welchen strategischen/technischen/organisatorischen Bereichen?
  - Welche unterschiedlichen Optionen gibt es?

# MEHR ZU REVIEWS

<https://www.heise.de/ratgeber/The-Art-of-Software-Reviews-Probleme-und-Risiken-zielsicher-identifizieren-4990332.html>