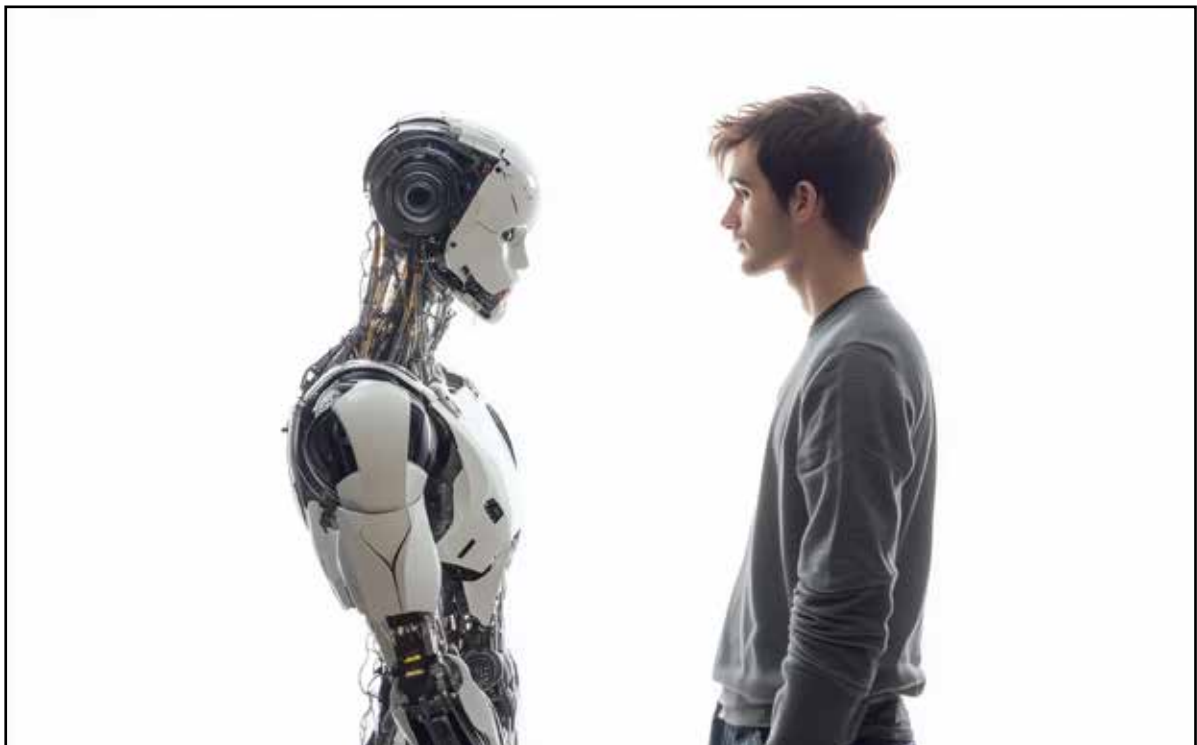


Letztes mal im Winter 2022!?

1



# Software Architectures for Enterprises SA4E

Peter Sturm  
Universität Trier  
Winter 2024



- Softwarearchitektur ~ Gebäudearchitektur



- Softwarearchitektur ~ Gebäudearchitektur
- Plan, Struktur, Infrastruktur



- Softwarearchitektur ~ Gebäudearchitektur
- Plan, Struktur, Infrastruktur
- Sieht einfach aus

11



- Softwarearchitektur ~ Gebäudearchitektur
- Plan, Struktur, Infrastruktur
- Sieht einfacher aus als es ist, Aller Anfang ist schwer

13





- Softwarearchitektur ~ Gebäudearchitektur
- Plan, Struktur, Infrastruktur
- Sieht einfacher aus als es ist, Aller Anfang ist schwer
- Enterprise = Horizontale und vertikale Skalierung



- Softwarearchitektur ~ Gebäudearchitektur
- Plan, Struktur, Infrastruktur
- Sieht einfacher aus als es ist, Aller Anfang ist schwer
- Enterprise = Horizontale und vertikale Skalierung
- Vorstellung/Konzept vs. Realität/Sachzwänge

17





- Softwarearchitektur ~ Gebäudearchitektur
- Plan, Struktur, Infrastruktur
- Sieht einfacher aus als es ist, Aller Anfang ist schwer
- Enterprise = Horizontale und vertikale Skalierung
- Vorstellung/Konzept vs. Realität/Sachzwänge
- Künstlerisches, Ästhetik, Baumeister

19



- Softwarearchitektur ~ Gebäudearchitektur
- Plan, Struktur, Infrastruktur
- Sieht einfacher aus als es ist, Aller Anfang ist schwer
- Enterprise = Horizontale und vertikale Skalierung
- Vorstellung/Konzept vs. Realität/Sachzwänge
- Künstlerisches, Ästhetik, Baumeister
- Modularität, Wartbarkeit, Erweiterbarkeit, Schnittstellen

21

# Software Architectures for Enterprises SA4E

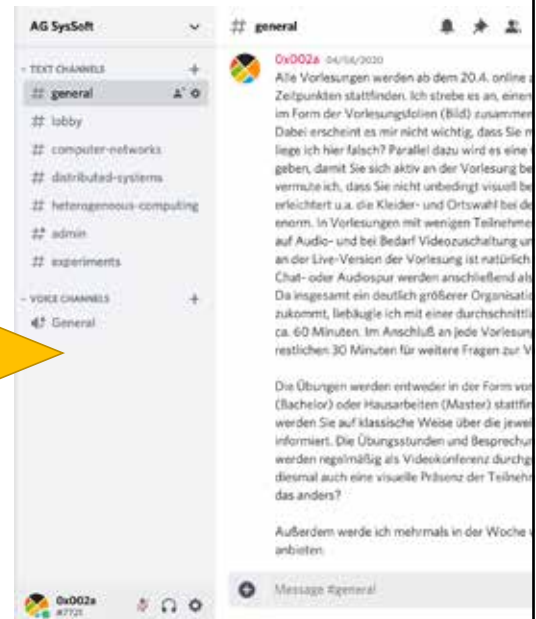
Peter Sturm  
Universität Trier  
Winter 2024

## Organisatorisches

24

## Kommunikationsformen

- Studip
  - Folienkopien
  - Übungsblätter
- Discord Server
  - Link in studip
- Email
  - [sturm@uni-trier.de](mailto:sturm@uni-trier.de)



25

## Übung (Portfolio)

- 3 Übungsblätter (jeweils 10 Punkte)
- 1 Abschlußprojekt (20 Punkte)
- Note ergibt sich aus der erreichten Punktzahl
- Keine Gruppenarbeit!

26

## Abschlußprojekt



## Tools

- Programmiersprachen
  - Java, Python, ...
  - Rust, C, C++, ...
- Integrierte Entwicklungsumgebungen
  - IntelliJ, (VSCode), ...
- Generative KI
  - Copilot, ...
- Versionierungssysteme
  - Git, Github
- Workflows
  - DevOps, ...



Noch Fragen?



29

# 1. Einführung

30



## Frage 1

- Wer hat schon eigene Software entwickelt?
- Umfang?
- Projektgröße in Anzahl Entwickler?

## Frage 2

- Wer hat schon fremde Software (wieder)verwendet?
- “Not invented here syndrom”?
- Umfang?



### Frage 3

- Wie wurde Software (wieder)verwendet?
- Code Scavenging?
- Bibliothek?
- Komponenten?
- Frameworks?
- Ganze Applikationen verklebt?



## Software Engineering

So ... Is it true that before coming here as a chief design engineer, you developed operating systems for PCs?

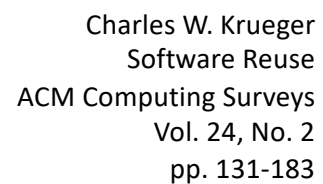
## Software Engineering

- 1968 NATO Software Engineering Conference
  - “Geburt” des Forschungsgebiets Software Engineering
- Softwarekrise wird erkannt
  - Entwicklung großer und zuverlässiger Systeme in kontrollierter und kosteneffektiver Form schwierig bis unmöglich
- Software Reuse
  - Zentraler Ansatzpunkt zur Überwindung der Krise
- 2024
  - Problematik immer noch nicht befriedigend gelöst?



## Software Reuse

Charles W. Krueger, 1992



## Artefakt – mehr als nur Sourcecode

- Architekturen
- Sourcecode
- Daten
- Entwürfe (Designs)
- Spezifikationen
- Dokumentation
- Abschätzungen (Estimates)
- „Human Interfaces“
- Pläne
- Anforderungen (Requirements)
- Testfälle
- „Prozesse“

## Techniken des Reuse

- Hochsprachen
- Bibliotheken
- Design- und Code-Scavenging
- Komponenten
- Softwareschema
- Generatortechniken
- Deskriptive Entwicklungssprachen
- Transformationssysteme
- Frameworks

## Techniken des Reuse

- Hochsprachen
- Bibliotheken
- Design- und Code-Scavenging
- Komponenten
- Softwareschema
- Generatortechniken
- Deskriptive Entwicklungssprachen
- Transformationssysteme
- Frameworks

## Bewertungskriterien

- Abstraktion
  - Beschreibung wiederverwendbarer Einheiten?
- Selektion
  - Lokalisierung, Vergleich und Wahl eines Artefakts
- Spezialisierung
  - Hohe Wiederverwendbarkeit bei generalisierten Artefakten
  - Spezialisierung nach erfolgter Selektion
- Integration
  - ... in ein vorhandenes bzw. zu entwickelndes System



## Reuse

- Ausgangspunkt
  - Entwicklungsprojekte sind keine einmaligen Ereignisse
- ... von Code
  - Frameworks
  - Komponenten
  - Produktfamilien
- ... von Erfahrungen
  - Gemeinsame Sprache: UML
  - Entwurfsmuster guter Softwarestrukturen
  - Erweiterte Softwareprozesse (z.B. CMM)

Warum diese  
Vorlesung?



44

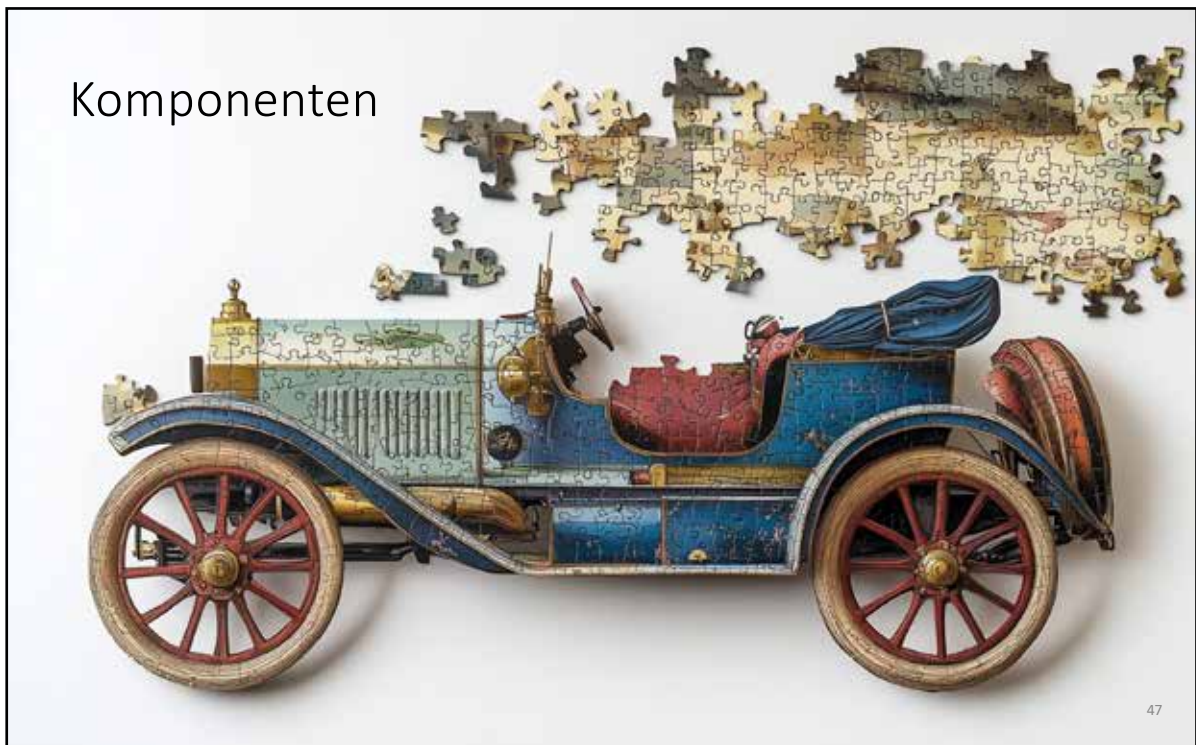
## Darum!

- Rechnernetze und TCP/IP sind bekannt
- Grundlagen “Software Engineering” sind bekannt
- Theorie der verteilten Systeme ist bekannt
- Konkrete Softwaretechniken und -architekturen
- Historischer Aspekt
  - Softwaresysteme sterben nicht aus!

45



46



## Komponenten

- Hohe Qualität der Einzelkomponente (Experten)
- Mehraufwand amortisiert sich durch Reuse
- Hoher Verbreitungsgrad (COM, EJB, .NET, CORBA)

## Frameworks



49

## Frameworks

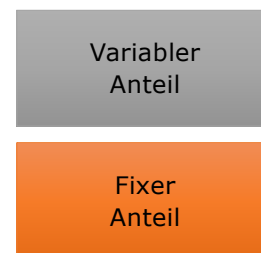
- Von Experten geschrieben  $\Rightarrow$  Hohe Qualität
- Je spezifischer die Anwendungsdomäne, desto besser
- Mehraufwand amortisiert sich durch Reuse





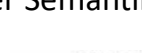
# Produktfamilien

- Abfolge von ähnlichen Produktentwicklungen
- Isolierung der variablen Produktanteile
- Instanzierbarer Entwicklungsprozeß

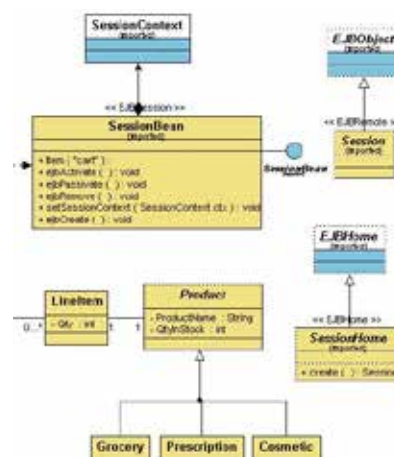


# Gemeinsame Sprache



- Graphische Notation mit wohldefinierter Semantik
  - Vielfältige Diagramme
    - Statisch / Struktur
    - Dynamisch / Ablauf
    - Einarbeitungsaufwand
  - Phasenspezifische Diagramme
- 
- ```

classDiagram
    class SessionContext {
        +SessionManager
    }
    class SessionManager {
        +SessionContext
    }
    SessionContext --> SessionManager : +SessionManager
    SessionManager --> SessionContext : +SessionContext
  
```
- The diagram shows two classes: SessionContext (blue) and SessionManager (yellow). SessionContext has a field SessionManager. SessionManager has a field SessionContext. There are two association arrows: one from SessionContext to SessionManager labeled +SessionManager, and one from SessionManager to SessionContext labeled +SessionContext.





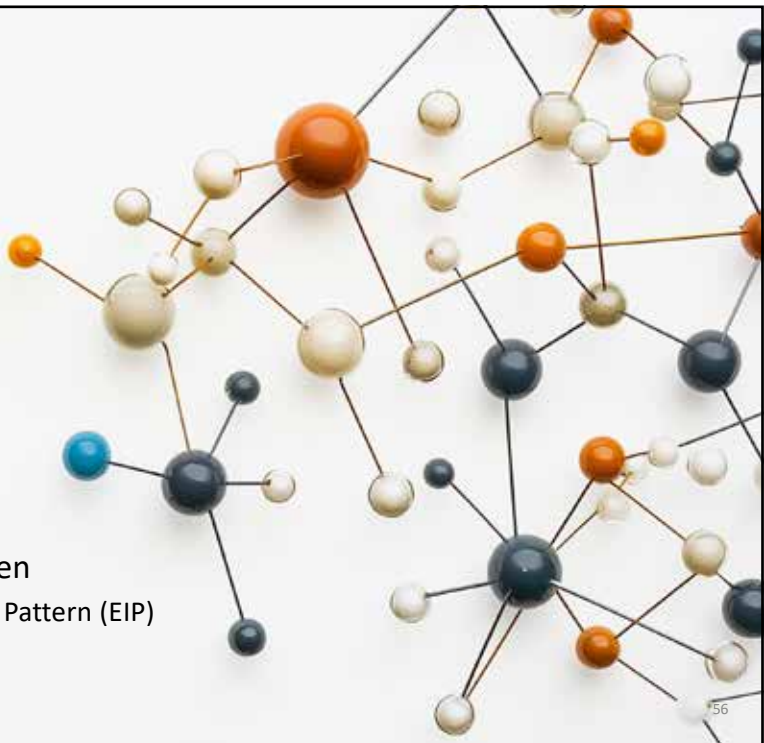
## Design Pattern



- Direkte Umsetzung in Form einer Code-Bibliothek nicht möglich
  - Struktureigenschaften der Anwendung
  - Verteilt sich über mehrere Codestellen
- Einheitliche Bezeichnung und Beschreibung
- Standardwerke, z.B. Gamma et al.

## Scope

- Innerhalb einer Anwendung
  - Design Pattern
  - Mikroskopisch
- Zwischen Teilen einer Anwendung
  - Mesoskopisch
- Zwischen Anwendungen
  - Enterprise Integration Pattern (EIP)
  - Makroskopisch



## Werkzeugunterstützung

- Tools sind unersetzlich geworden
  - Dramatische Steigerung der Effizienz bei hoher Qualität
  - Moderne Werkzeuge unterstützen höhere Ebenen (z.B. Design Pattern)
  - Refactoring
  - Hoher Einarbeitungsaufwand notwendig
- Wiederverwendung von Erfahrung

**A fool with a tool is still a fool**

Neu oder Alt?







Aus Alt mach Neu





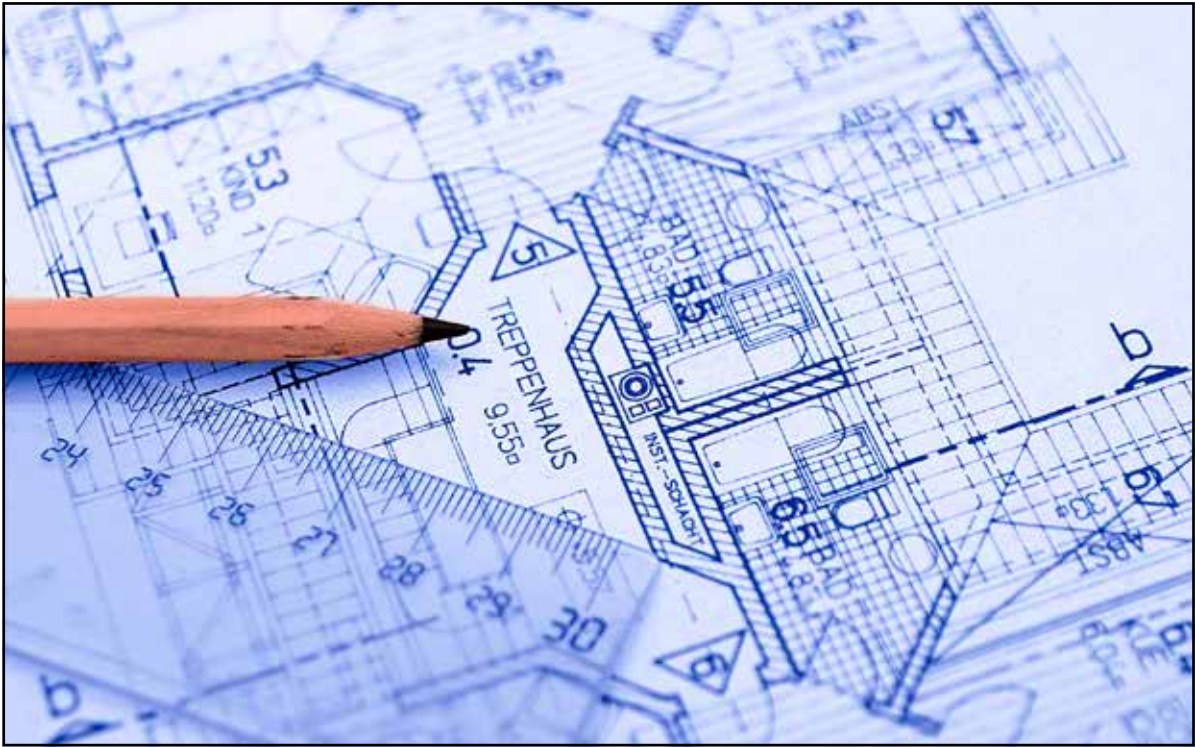


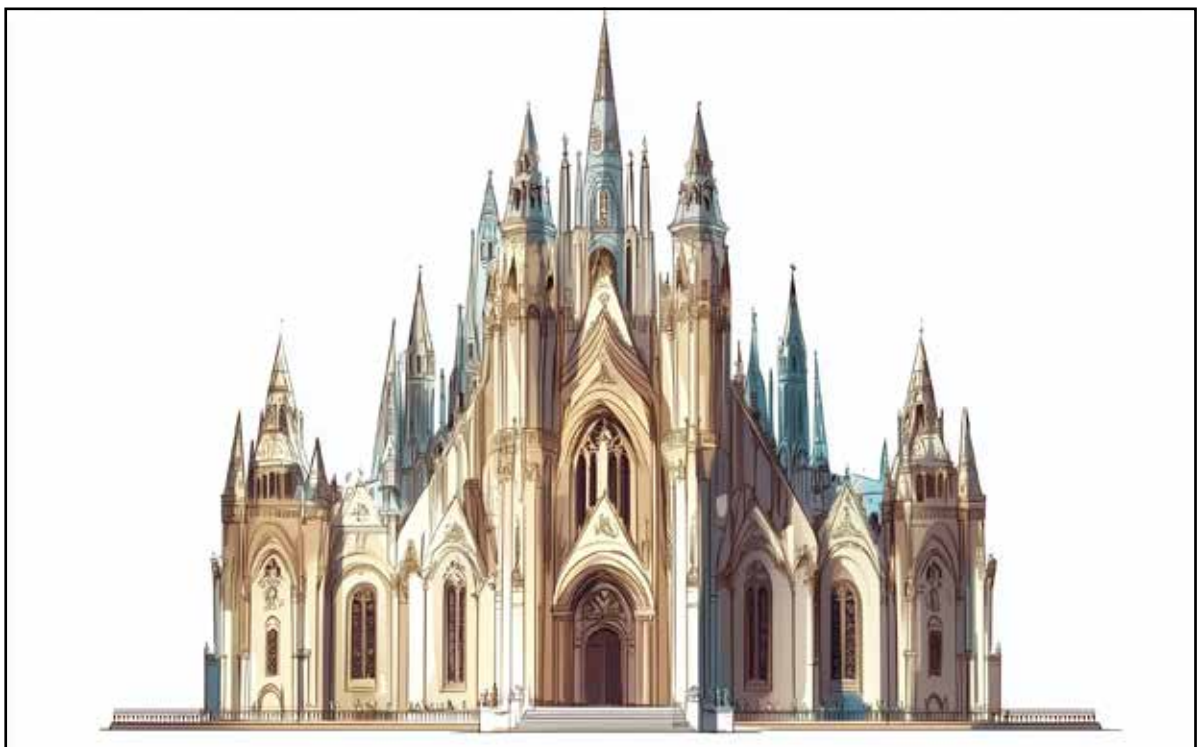


## EIP

- Enterprise Integration Patterns
- Integrationsstil
- Datenrepräsentation
- Routing
- Processing

## Architecting





# Inhalte









## Was kommt?

- Altes
  - RPC, COM, ActiveX, CORBA, .NET, EJB
- Modernes
  - SOA, Web Services, RESTful, Microservices
  - Container(gruppen) – Kubernetes u.a.
- EIP
  - Apache Camel
- Architekturen / Architecting
  - Facebook, Netflix, Uber, Spotify, ...



## Übung (Portfolio)

- 3 Übungsblätter (jeweils 10 Punkte)
  1. UDP, TCP, RPC, RMI, Remoting (Mitte November)
  2. Moderne Sprachansätze, Apache Camel (Mitte Dezember)
  3. DI, Inversion of Control, Spring Boot (Mitte Januar)
- 1 Abschlußprojekt (20 Punkte)
- Note ergibt sich aus der erreichten Punktzahl
- Keine Gruppenarbeit!

81

## Abschlußprojekt



