Adaptive Software Architekturen Labor

SS2024

Roland J. Graf / Henrik Binggl

Roadmap

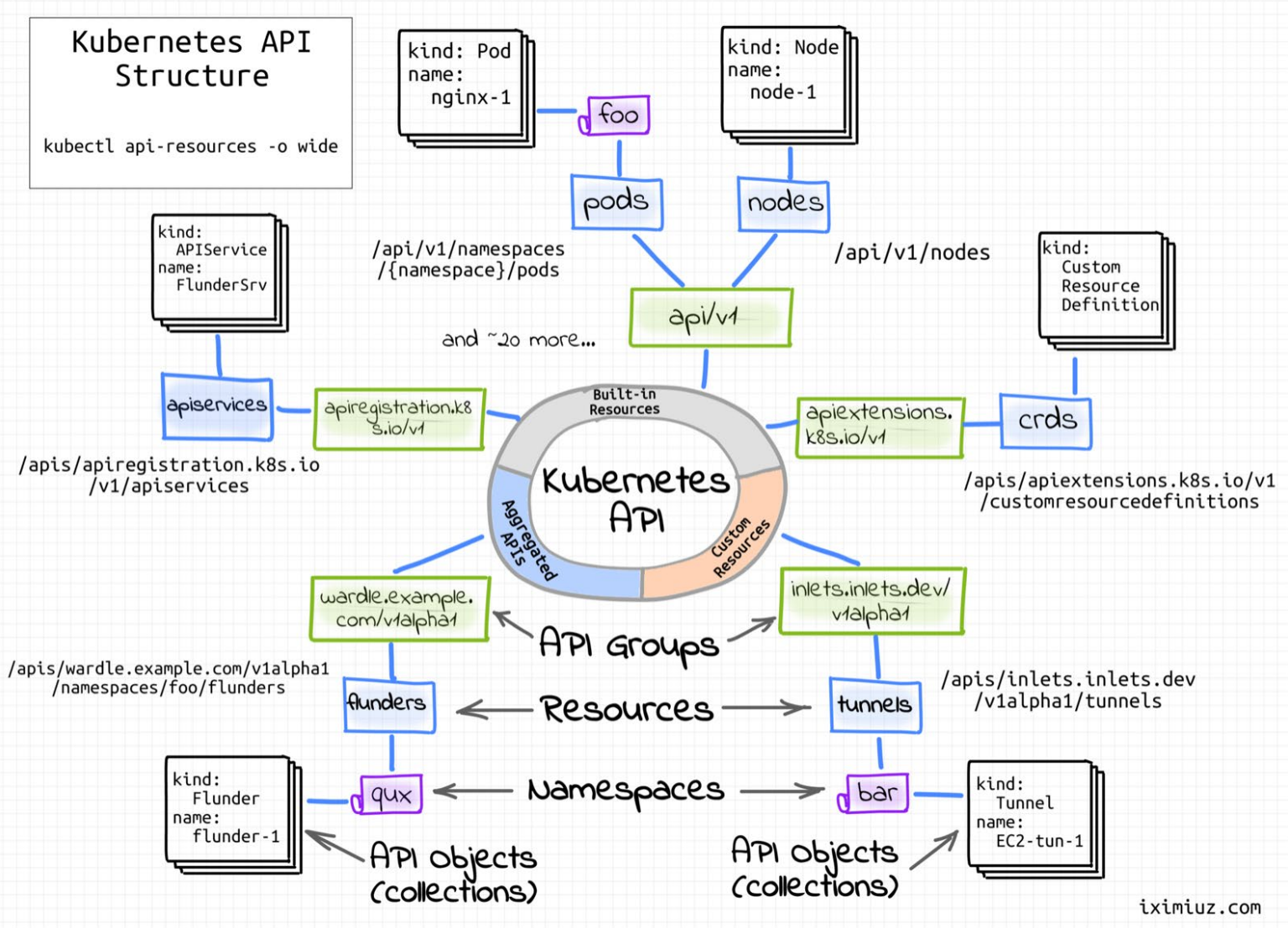
- LB01 - Architekturentwurf
  (Prinzipien, Techniken, Muster)
- LB02 - Architektur-Präsentation und –Dokumentation
  (Praxisregel, Frameworks, Templates)
- LB03 - Architektur-Reviews
  (Bewertung, Qualitätsmerkmale, Metriken)
- LB04 - Abschluss

Roadmap

- LB01 - Architekturentwurf
  (Prinzipien, Techniken, Muster)
- LB02 - Architektur-Präsentation und –Dokumentation
  (Praxisregel, Frameworks, Templates)
- LB03 - Architektur-Reviews
  (Bewertung, Qualitätsmerkmale, Metriken)
- LB04 - Präsentation & Feedback

Was ist Softwarearchitektur?

Was ist Softwarearchitektur?



Was ist Softwarearchitektur?

- *"Die Softwarearchitektur beschreibt bis zu einem gewissen Detaillierungsgrad die **Struktur** (Schichtung, Modularisierung etc.) und die **Systematik** (Pattern, Konventionen etc.) eines Softwaresystems.„* [Stahl, 2005]
- *„Es geht um die **Zerlegung eines Systems**… Es geht um schwer zu ändernde Entscheidungen … Architektur bezeichnet alles, was wichtig ist.“* [Fowler, 2003]
- *”Architektur ist die grundlegende Organisation eines Systems, bestehend aus den Komponenten, ihren Beziehungen untereinander und zur Umgebung sowie den Prinzipien die Design und Entwicklung leiten*” [IEEE 1471-2000] Institute of Electrical and Electronics Engineers (IEEE), https://www.ieee.org/

Was ist Softwarearchitektur?

- *”… not all design is architecture. Architecture represents the significant design decisions that shape a system, where significant is measured by cost of change“*
- Es geht um die **Zerlegung eines Systems**… Es geht um schwer zu ändernde Entscheidungen … Architektur bezeichnet alles, was wichtig ist. [Fowler, 2003]
- ”Architektur ist die grundlegende Organisation eines Systems, bestehend aus den Komponenten, ihren Beziehungen untereinander und zur Umgebung sowie den Prinzipien die Design und Entwicklung leiten” [IEEE 1471-2000] Institute of Electrical and Electronics Engineers (IEEE), https://www.ieee.org/

Softwarearchitektur als Disziplin

Primäre Aufgabe

- Zerlegung des Systems in Hauptbestandteile auf oberster Ebene (Teil des System-/Softwareentwurfs)
- Festlegen derer Beziehungen
- Verständlich machen der Systemstrukturen

Effiziente Entwicklung komplexer Software benötigt die **Definition** von

- Vorgehensweisen
- Methoden, Muster und Schablonen
- Werkzeuge

Ziel des Architekturentwurfs

- **Konzeption** eines Bauplans des Systems
- unter Orientierung an der Erstellung, Wartung, Pflege und Weiterentwicklung des Systems.

- **Architekturbeschreibung** dient als
- Kommunikations- und Abstimmungsplattform,
- Design-, Implementierungs- und Ablaufrahmen

Vorgehensweise beim Architekturentwurf

- Erarbeitung und Vereinbarung geeigneter Abstraktionslevel
- Identifizieren, Priorisieren und Zuordnung von funktionalen und nicht-funktionalen Anforderungen
- Berücksichtigen existierender Lösungen, Teilsystemen & möglichen Alternativen
- Festlegen der Komponenten und Schnittstellen und deren Zusammenspiel
- Evaluierung und Einbeziehung von spez. Techniken & Vorgehensweisen
- Erstellen von Prototypen bzw. Referenzimplementierungen (optional)
- Einführen (und Überwachen) von Implementierungsrichtlinien
- Ableiten von Konsequenzen für die Projektplanung und -organisation

Architekturentwurf - Prinzipien

- Einfachheit gewinnt (Keep It Small and Simple, **KISS**)
  - https://en.wikipedia.org/wiki/KISS_principle
- Vermeide Redundanz (Don‘t Repeat Yourself, **DRY**)
  - https://en.wikipedia.org/wiki/Don%27t_repeat_yourself
- Fokus auf die Anforderungen (You Aren‘t Gonna Need It, **YAGNI**)
  - https://www.martinfowler.com/bliki/Yagni.html

- **SOLID**-Prinzipien
  - https://dev.to/moresaltmorelemon/do-me-a-s-o-l-i-d-4g93

Impuls: Warum "zukunftssichere" Architekturen gefährlich sind | heise Developer

Wege zu einer besseren/guten Architektur

- Immer so einfach wie möglich (aber nicht einfacher)
- Redundanzvermeidung, aber nicht auf Kosten der Verständlichkeit.
- Trennung von Verantwortlichkeiten, Schnittstellen & Implementierungen
- Nicht möglichst genial, sondern immer möglichst verständlich
- Bei Unklarheiten/Unsicherheiten fragen
- Beginnen Sie möglichst "*hardest first*" - der Rest ist Fleißarbeit.
- Unterscheidung von Prototypen & Produktionsumgebungen
- Entwerfen Sie, was Sie auch selbst gerne entwickeln würden
- Es gibt keine Spezialfälle, sondern immer nur unvollkommene Modelle.

Wege zu einer besseren/guten Architektur

Einhaltung von **Grundprinzipien (Best practices)**

- Trennung der Zuständigkeiten (Separation of Concerns)
- Vermeidung von Abhängigkeiten (Lose Kopplung)
- Datenabstraktion & Geheimnisprinzip

Wesentlicher erster Schritt bzgl. **Aufteilung**

- Trennung von fachlichen und technischen Aspekten

Überwachung

- Erfüllung einer "guten" Architektur während der gesamten Realisierungsphase
- Architekturmanagement
- Mögliche Unterstützung durch entsprechende Werkzeuge

Architektur – Aspekte von Software

Durch die Architektur beschriebene Aspekte von Software

- Bausteine, Verantwortlichkeiten, Beziehungen, Interaktionen, Verhalten zueinander und nach außen
- Darstellen von verschiedenen Strukturen (Teilarchitekturen) durch architektonische Sichten (Views), z. B. Facharchitektur, Verteilungsstruktur, funktionale Schichtung, usw.
- Typische Bausteine: Klassensysteme, Frameworks, Module, Komponenten, Subsysteme
- Existierende und mögliche Abläufe im System

Architekturgrundsätze & -randbedingungen

Grundsätze

- Funktion
- Qualität
- Stil und Ästhetik

- Anforderungserfüllung
  - funktionale Anforderungen
  - nicht-funktionale Anfordg.

Randbedingungen

- Entwicklungszeit
- Kosten & Rendite
- Anforderungen des Auftraggebers
- Benutzer des Systems
- Hard- und Softwareumgebung
- Wiederverwendung
- Erfahrung des Entwicklungsteams

Architekturdarstellung als Abstraktion

- Gesamtarchitektur ist keine Darstellung des Innenlebens der Bausteine (=> Blackboxes)

  Ausnahmen: signifikante Auswirkungen auf das Verhalten des Gesamtsystems zu erwarten

- Grad der Detaillierung für Subbausteine wird von Qualitätsvorgaben und zugehörige Risiken bestimmt (=> Grey-/Whiteboxes)

SW-Architektur / Qualitätsanforderungen

- … im Nachhinein of nur mehr schwer beeinflussbar (teuer)
- … beeinflussen grundlegende Strukturentscheidungen
- … “querschnittlich” (für viele bis alle Projektmitglieder relevant)

- **Qualitätsanforderungen** sind die wichtigsten Anforderungen für Architekturarbeit
- **Qualitätsmerkmal** ist eine Eigenschaft einer Software, die sich bei der Erstellung, Benutzung oder Weiterentwicklung zeigt
- **Qualitätsziele** sind die wichtigsten geforderten Qualitätsmerkmale für eine Software-System

SW-Architektur / Qualitätsanforderungen

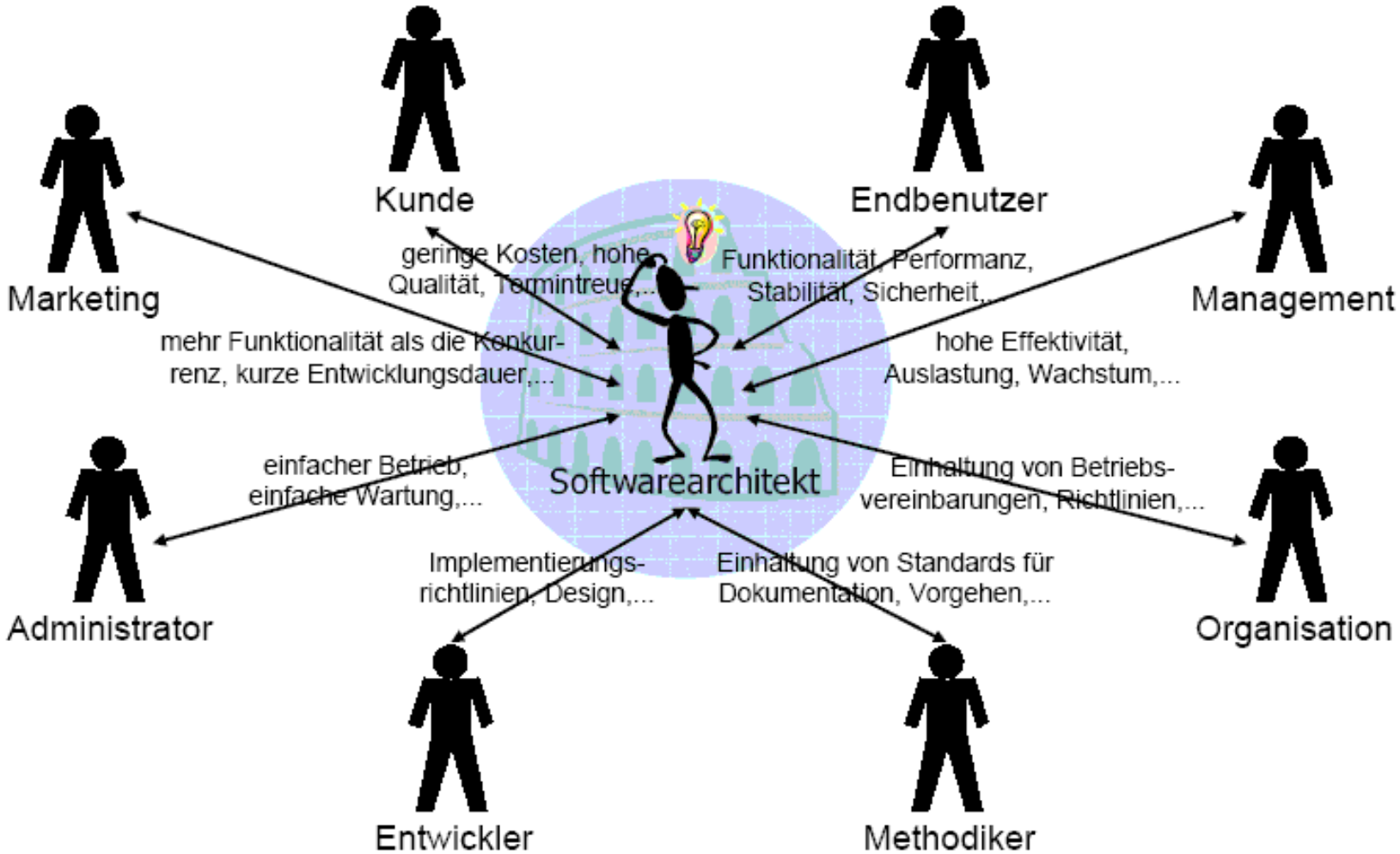
Harte Kriterien

- Performanz
- Sicherheit
- Verfügbarkeit und Zuverlässigkeit
- Robustheit
- Funktionsumfang
- Benutzbarkeit

Weiche Kriterien

- Testbarkeit, Integrierbarkeit
- Wartbarkeit
- Änderbarkeit
- Portierbarkeit
- Skalierbarkeit
- Wiederverwendbarkeit

Architekt „…*in the Middle*“?



Aufgaben der Softwarearchitekt:innen

- Er/sie ist "Vordenker" mit der **Sicht auf das große Ganze**
- Herausarbeiten, **Kommunikation** und **Diskussion** der Architektur
- Herbeiführung notwendiger **Entscheidungen** im Team
- **Planen**, **Überwachen**, **Moderieren** & **Unterstützung** der Umsetzung
- Einbringung geänderter Anforderungen, Umstände oder gewonnener Erkenntnisse => **Change Management**
- **Abstimmung** mit Stakeholder
- **Vermittlung** von Technologie-Konzeption

Anforderungen an Softwarearchitekt:innen

- Analytisches Denken und Abstraktionsfähigkeit
- Umfassendes theoretisches und praktisches Know-How bzgl. aktueller Technologien, Methoden, Werkzeuge, Architekturstile, Realisierungs-möglichkeiten
- Viel Projekt- und vorzugsweise auch Entwicklungserfahrung
  Aber: Ein Architekt muss aktuell nicht kodieren (umsetzen) können
- Benötigt Kommunikations-, Überzeugungs- & Integrationsfähigkeit

Achtung: Der Softwarearchitekt muss nicht jede kleine Design- und Implementierungsentscheidung absegnen!

Architektur & Pattern

Patterns auf den Ebenen: Architektur, Design und Implementierung

General **R**esponsibility **A**ssignment **S**oftware **P**atterns (GRASP)

- keine Patterns im herkömmlichen Sinn (wie GoF),
  vielmehr Beschreibungen von Vorgehensweisen
  Adressieren die Fragestellung: Wer ist wofür verantwortlich/zuständig

Beispiele:

- High Cohesion
- Low Coupling
- Information Expert
- Creator
- Creator,
- Controller
- Polymorphism
- …

Typische Architektur-Pattern

Allgemeine Pattern: Pattern Oriented Software Architecture (POSA)

- Layers (OSI Model)
- Pipes & Filters (Unix Philosophy)
- Blackboard (Data Centered Architecture)
- Broker, Service Oriented Architecture (SOA)
- Client-Server (3-tier Architecture)
- Model-View-Controller, MVVM (GUI Applications)
- Microkernel / Monolith
- MicroServices (REST-API)

- Architektur ist ein “Kind der Zeit”
  - Architektur-Trends?
  - Modeerscheinung?
- Architektur wird von Technologie beeinflusst
- https://www.infoq.com/articles/architecture-trends-2022/