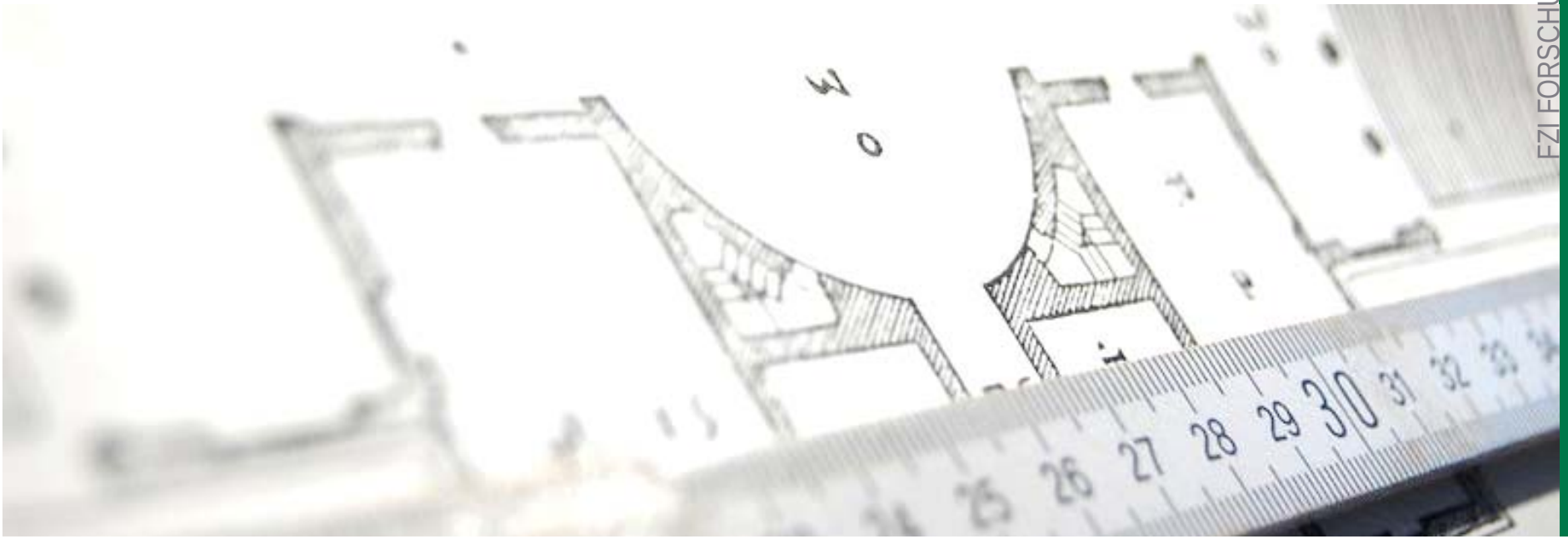
Architekturentscheidungen in der agilen Entwicklung

Dr.-Ing. Klaus Krogmann
FZI Forschungszentrum Informatik
Software Engineering



FZI FORSCHUNGSZENTRUM INFORMATIK

„Zwitscher“ Kurznachrichtendienst

„Zwitscher“ Kurznachrichtendienst für die unternehmensinterne Kommunikation aller Karlsruher Standorte des Unternehmens

Webbrowser

HTML Front End

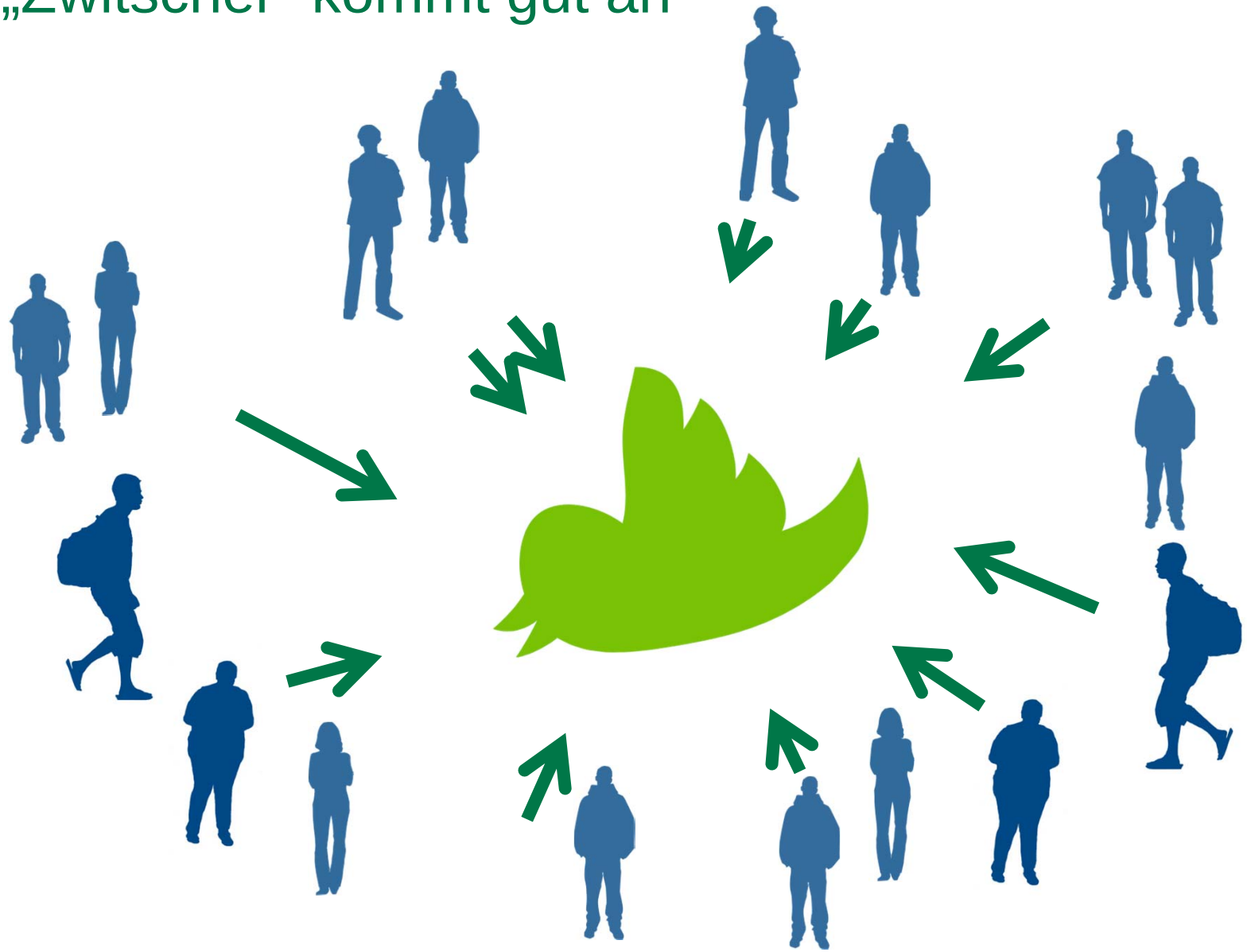
Logikkern mit Transaktionsverwaltung

Datenbank

Entwicklung mit
- Applikationsserver
- Relationaler Datenbank

Weil
- Gute Erfahrung
- Best Practice und schnelle Lösung

„Zwitscher“ kommt gut an



„Zwitscher“: Skalierung macht keinen Spaß

- Massiver Umbau der Architektur
  - Skalierung
  - Cloud-Betrieb
- Weltweite Verteilung
  - Replikation
    → wie Datenbestand verwalten?
  - Datenkonsistenz
    → wie Datensynchronisation?
  - 24/7 weltweite Verfügbarkeit
    → wie aktualisieren?

Was ist bei „Zwitscher“ passiert?

- Anforderungen / Wachstum unklar
  → vielleicht

- Aber
  - Implizite Architekturannahmen
  - Keine geprüften Architekturentscheidungen
  - Verwendung des „das haben wir schon immer so gemacht“-Technologiestack
- Und
  - „Durch unsere Agilität können wir die Architektur jederzeit durch ein Refactoring ändern“
  - Agiles Anti-Pattern: Planungsverweigerung [1]

[1] Klaus Krogmann, Matthias Naab, and Oliver Hummel. Agile Anti-Patterns - Warum viele Organisationen weniger agil sind, als sie denken. *JAXenter Business Technology*, 2.14:29-34, June 2014.“**, Klaus Krogmann, Matthias Naab und Oliver Hummel**

Thesen

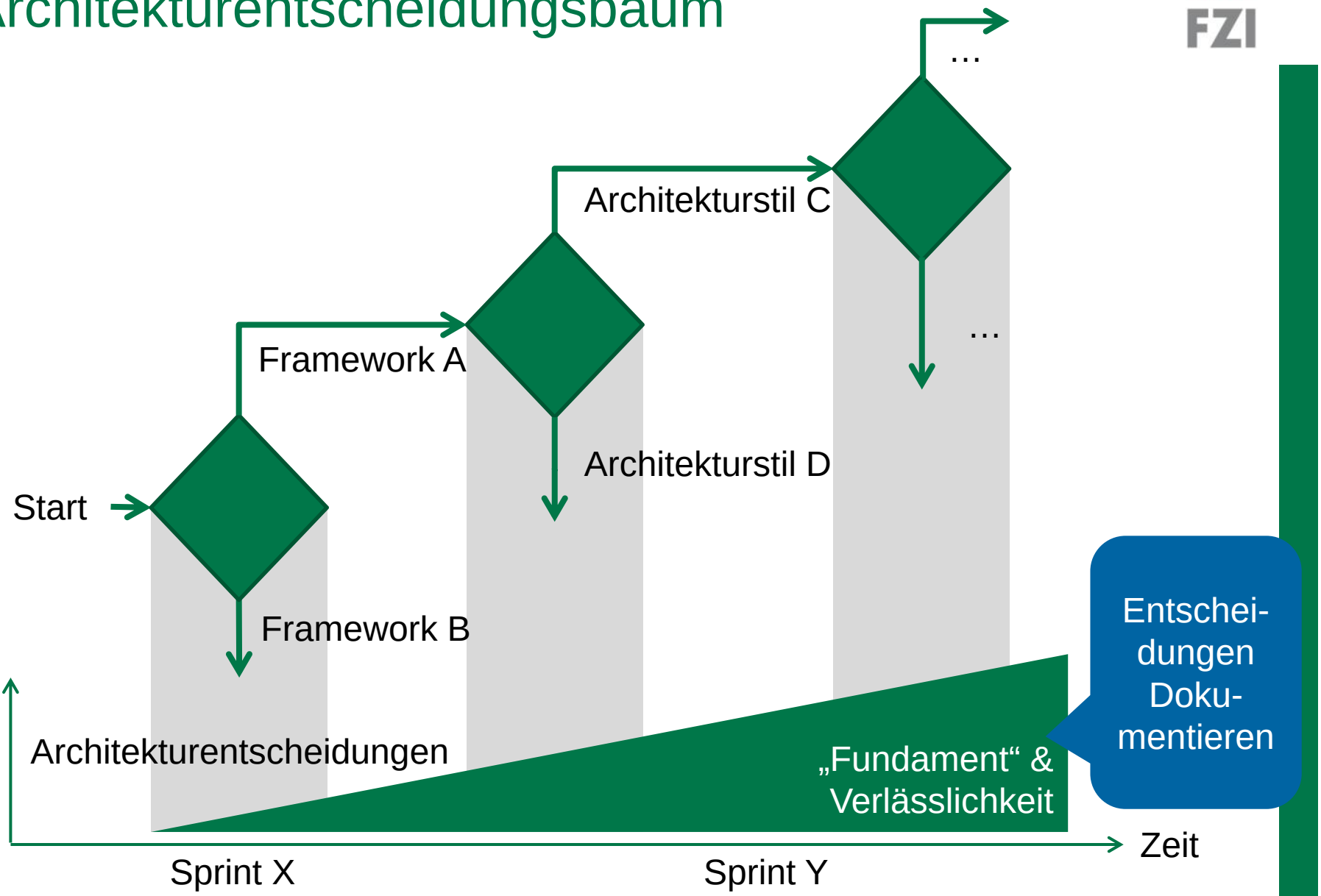
These 1

Jede Software-Entwicklung — egal wie agil — muss Entscheidungen zur Software-Architektur treffen

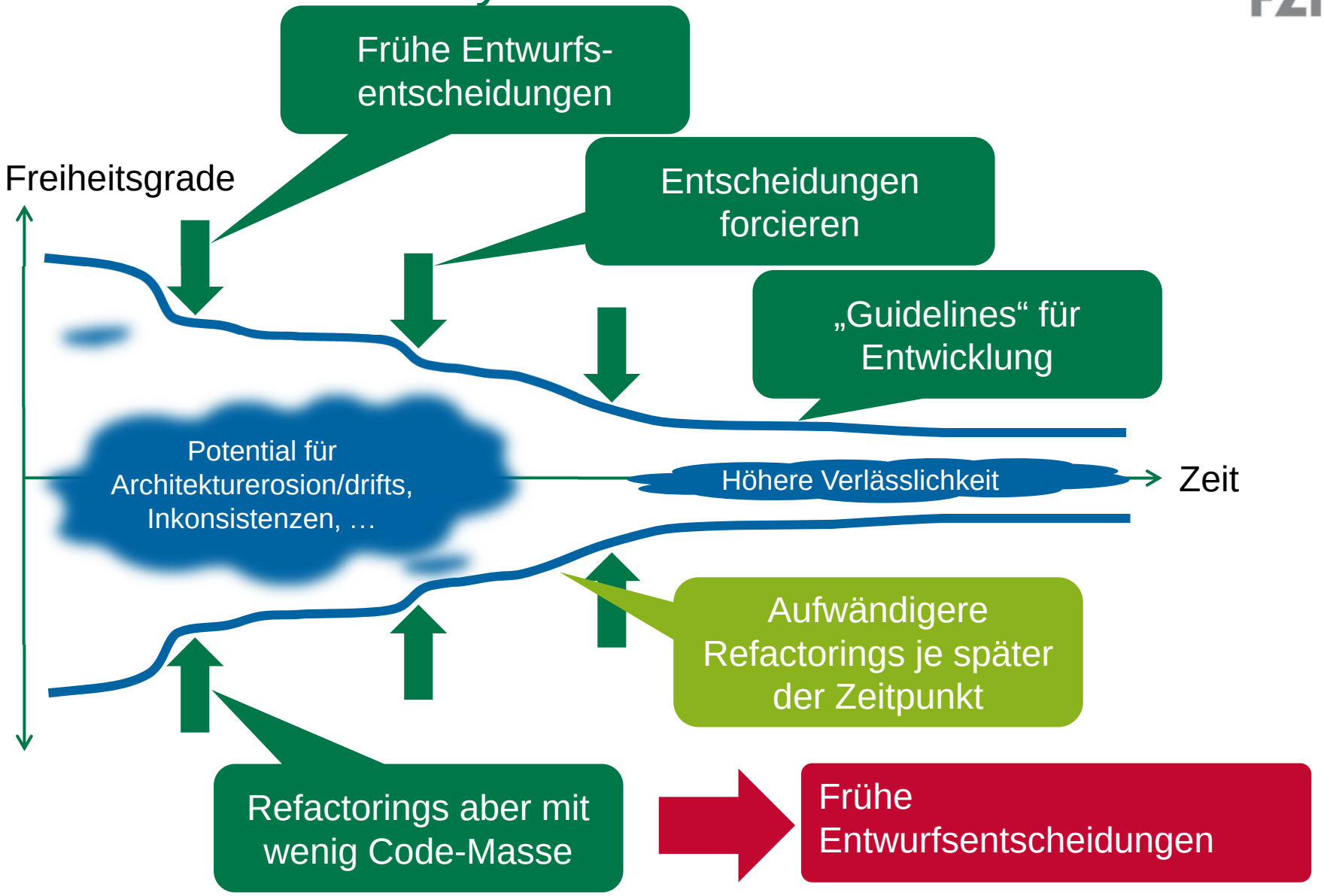
These 2

Architekturentscheidungen schränken die Freiheitsgrade ein, sind aber zugleich ein Stützpfeiler der Entwicklung

Architekturentscheidungsbaum

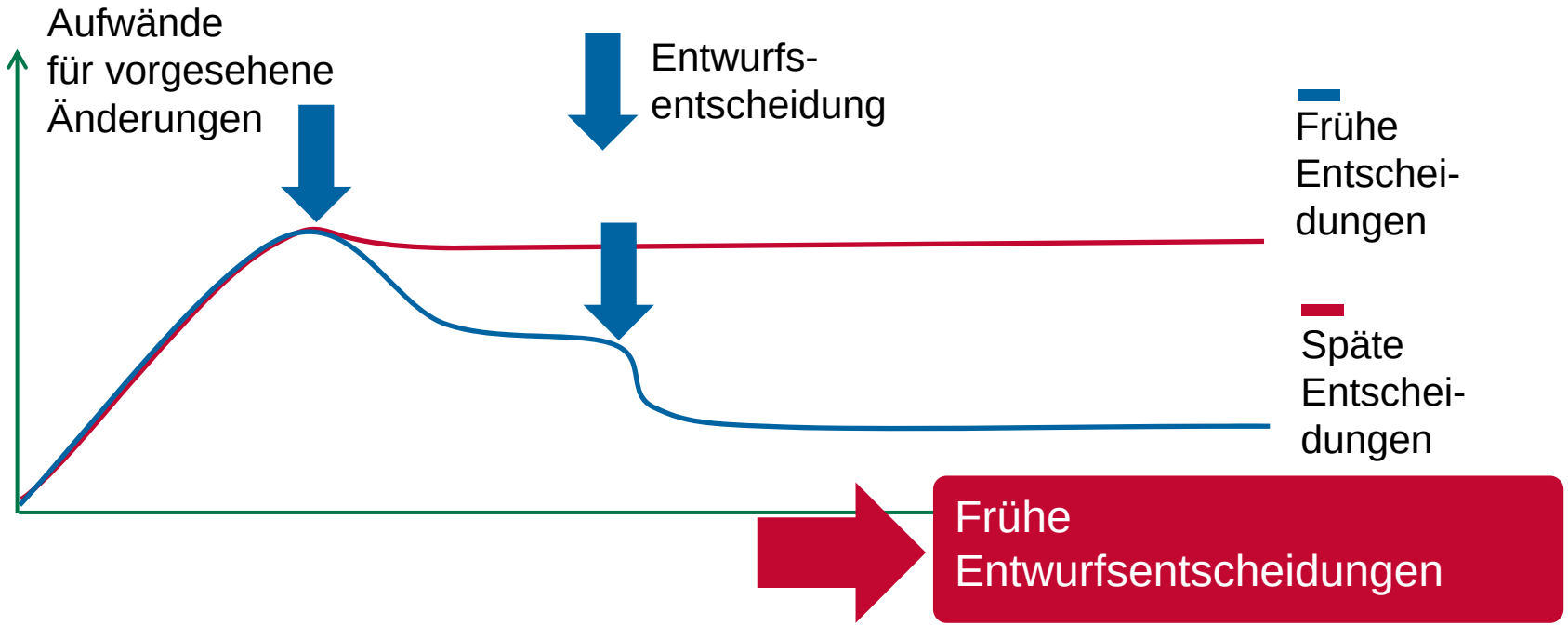


Cone of Uncertainty



Frühe vs. späte Architekturentscheidungen

- „Kill it with iron“ / SOA / Cloud / …
  kostet und setzt Qualitätsbasis (bspw. Skalierbarkeit) voraus
- Architekturentscheidungen ermöglichen vorgesehene Änderungen mit weniger Aufwand zu realisieren
- *Fehlende* Architekturentscheidungen lassen ungewollten Freiraum, der Aufwand für Refactorings & Co. steigert

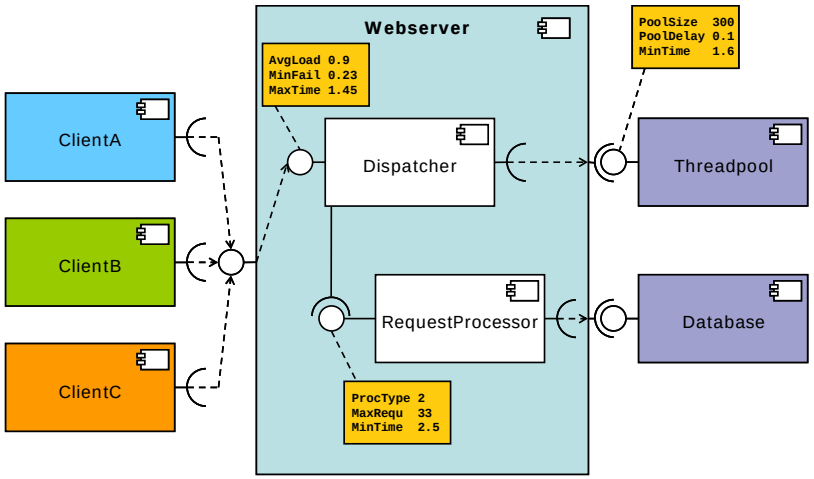


Frühe Entwurfsentscheidungen?

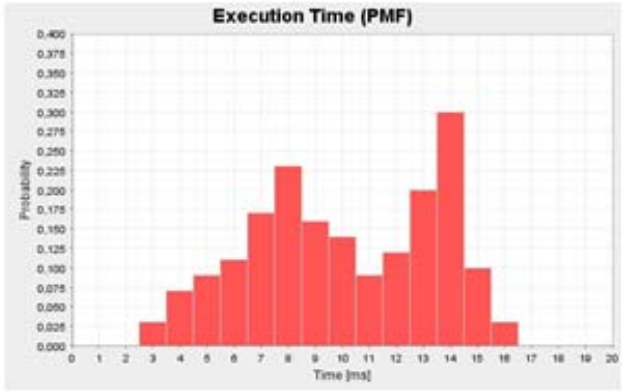
- Architektur noch offen
- Anforderungen nicht fix
- Code nur teilweise da
- Bestandssysteme müssen angeschlossen werden

→ **Geht das?**

Modellbasierte Architekturanalyse



Analyse

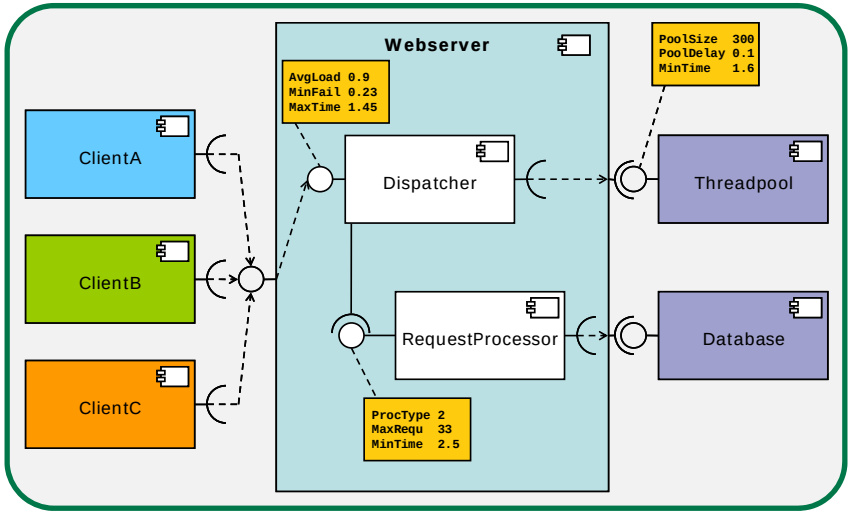


Modell
einer komponentenbasierten
Software-Architektur

- Ausführungszeit
- Durchsatz
- Ressourcenauslastung
- Verfügbarkeit
- …

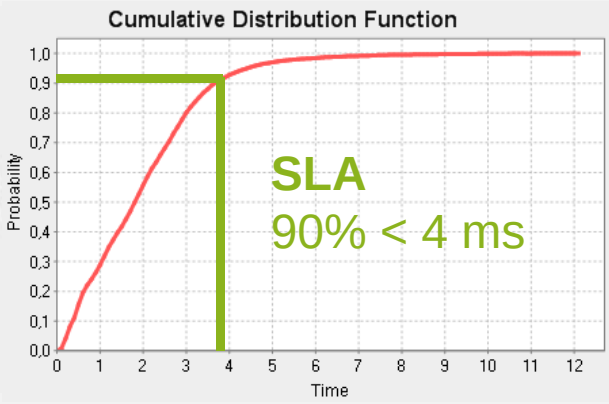
Vorgehen bei modellbasierten Architekturanalysen

Systemmodell



Analyse

Vorhersage



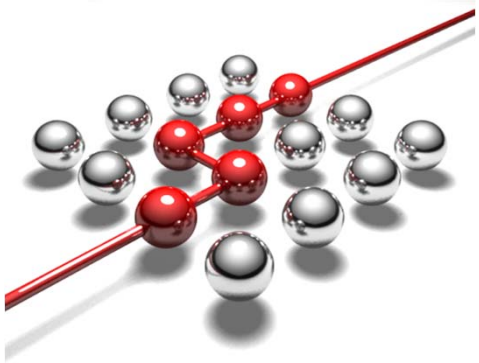
Entscheidungs-findung

Realisierung der Lösung mit zufriedenstellenden Qualitätseigenschaften

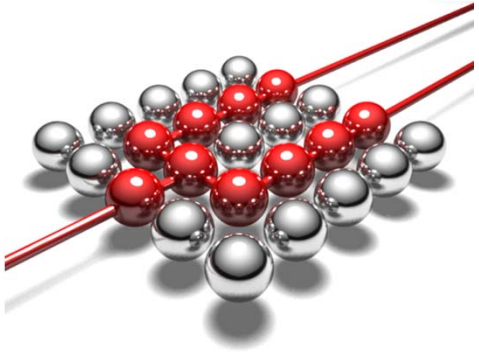
Feedback

Verfeinerung/ Änderung/ Alternativenerprobung

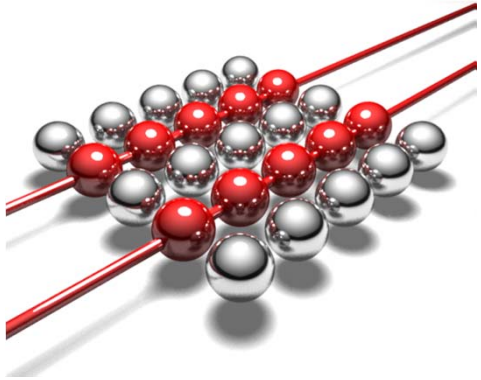
Szenarienbasierte Untersuchung



Entwurfsalternativen

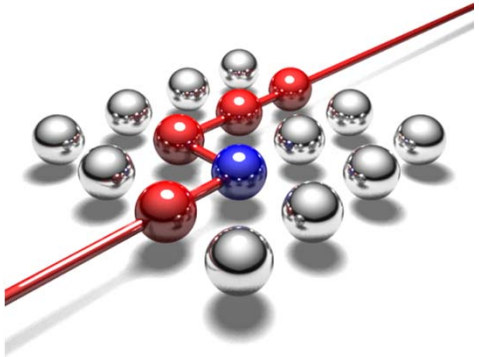


Sizing



Skalierbarkeit

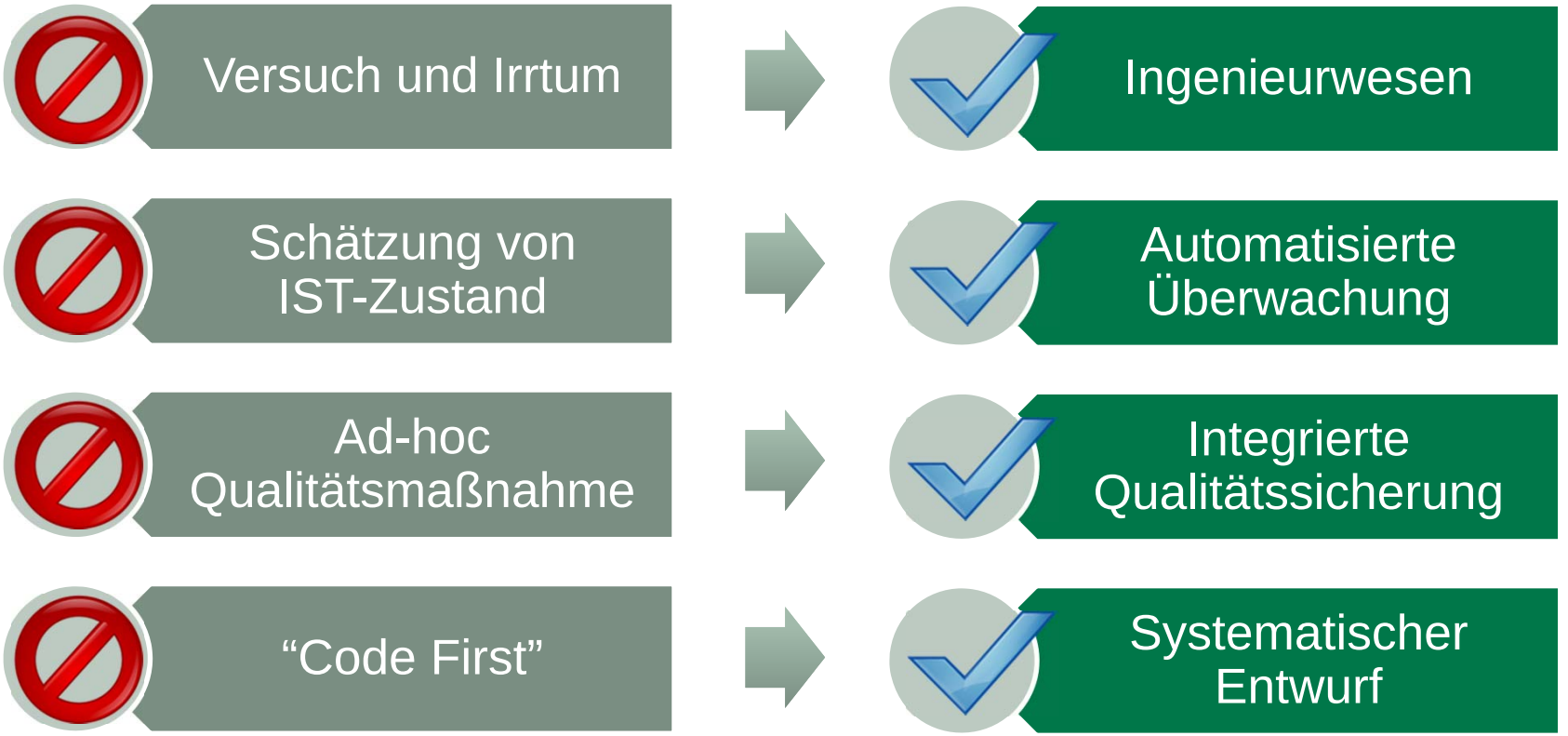
Optimierte Ressourcennutzung



Konfigurations-optimierung

Erweiterung von Bestandssoftware

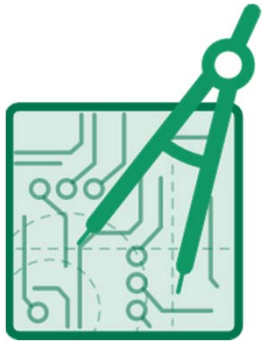
Architekturentscheidungen dank Ingenieurwesen



Vorhersagbare Eigenschaften & Gezielte Konstruktion

Vorhersagbare Eigenschaften

- Antwortzeitverhalten
- Skalierbarkeit
- Ressourcenausnutzung
- Zuverlässigkeit / Verfügbarkeit
- Recovery-Fähigkeiten
- Failover-Verhalten



Systeme und Systemlandschaft

- Gesamtsystemverhalten auf der Architekturebene
- Verhalten von Kompositionen
- Auswirkungsanalyse

Zielgerichtete Konstruktion

- Qualitative und quantitative Entscheidungsgrundlage
- Robustheit / Skalierbarkeit / Effizienz
- Erweiterbarkeit

Werkzeuge und Ansätze

- Palladio Software Architecture Simulator
- SimuLizar
- QPME: Queueing Petri net Modeling Environment
- SPE·ED Performance Modeling Tool
- Carleton LQ Tools: Layered queueing network solver and simulator
- Java Modelling Tools: Multiple performance models
- Delsi 2.0: Discrete-event simulation system for queuing systems
- LINE solver for queueing network models

- PRISM: System Reliability Center
- PRISM model checker: Analysis of systems with probabilistic behaviour
- Reliasoft: Reliability analyses and predictions
- ALD Reliability and Safety Tools
- Windchill Quality Solutions: “Relex”

Indikatoren für genaue Architekturevaluation

- **Projektrisiken** aus Qualitätssicht
- Neue Technologien und Architekturkonzepte
- Neue Domäne
- Neue Benutzergruppen
- Zusicherung strikter SLAs

Art der Architektur

„Standard-Architektur“

- + Ähnliches Projekt abgewickelt
- + Domäne ist bekannt
- + Erfahrung mit Technologie besteht
- + Auswirkung von Anforderungen bekannt

- → Referenzarchitekturen
- → Standardtechnologien
- → Best Practices
- → Fühlt sich agil an

Individuelle Architektur

- + Projektrisiken aus Qualitätssicht
- + Neue Technologien
- + Neue Domäne

- → Quantitative Architekturbewertung
- → Technologievergleich
- → Architekturspezifische Guidelines
- → allg. Best Practices

Neigung zu impliziten Architekturentscheidungen oder Referenzarchitekturen
- Framework- und Technologievorgaben
- Mitarbeiterwissen und -erfahrung

Architekturentscheidungen und Agilität

Agile Entwicklung

- Nutzt Freiheitsgrade der Architektur
- Pflegt und entwickelt Best Practices weiter
- Mehr Agilität (Feature je Zeit) möglich

Architekturentscheidungen

Q: Wie früh?
A: So früh wie möglich (aber nicht früher)

Q: Was ist agil änderbar?
A: Alles innerhalb der gegebenen Leitlinien

Q: Welche Teile sollten früh stehen?
A: Grobarchitektur mit bekannter Eignung für Anforderungen

Q: Wie kann man Referenzarchitekturen nutzen?
A: Als Wiederverwendungsmöglichkeit für bekanntes Terrain

Q: Welche Unterstützung gibt es für frühe Architekturentscheidungen?
A: Modellbasierte Verfahren (helfen auch beim Verstehen)

Thesen

- **These 1:** Jede Software-Entwicklung — egal wie agil — muss Entscheidungen zur Software-Architektur treffen
  - Auch implizite Entscheidungen schaffen eine Architektur
  - Implizite Architekturentscheidungen explizit machen → Kommunikation

- **These 2:** Architekturentscheidungen schränken die Freiheitsgrade ein, sind aber zugleich ein Stützpfeiler der Entwicklung
  - Erhöhte Verlässlichkeit für Entwickler
  - Gemeinsame Systemvorstellung gefördert

Kontakt

Dr.-Ing. Klaus Krogmann
FZI Forschungszentrum Informatik
Bereichsleiter Software Engineering
Haid-und-Neu-Str. 10-14 | 76131 Karlsruhe

krogmann@fzi.de | http://www.fzi.de/se
+49 721 9654 636 (fon | gesch.)
+49 721 9654 637 (fax | gesch.)