

Kollaborative Architektur-Reviews

Stefan Toth

Software Architecture Forum, 17.06.2026





Warum Reviews?

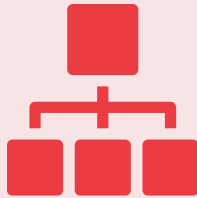
Kostet etwas und schafft keinen direkten Wert...



Architektur-Themen

Zerlegung

Welcher Architekturstil?
Wie zerfällt die Anwendung?
Teilsysteme, Module,
Komponentenbildung,
Abhängigkeiten ...



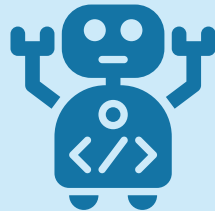
Zielumgebung

Wo läuft die Software?
Beim Endanwender, im
eigenen Rechenzentrum,
Cloud, Verteilung,
Virtualisierung ...



Technologie-Stack

Was setzen wir ein?
Programmiersprache(n)
Bibliotheken, Frameworks,
Middleware,
Querschnittsthemen

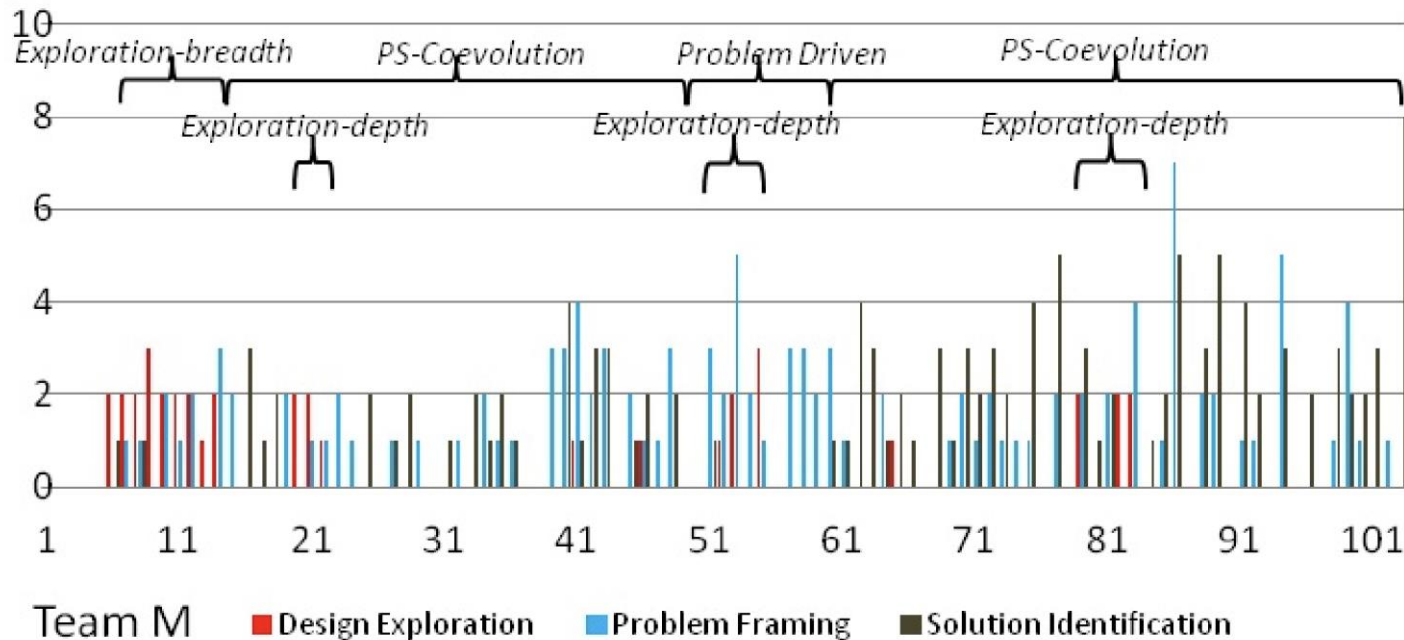


Vorgehen

Wie arbeiten wir?
Planen, Entwickeln, Testen,
Bauen, Dokumentieren,
Ausliefern, Nachjustieren, ...



Selbst Top-Down Design ist unübersichtlich



Aus: "Decision Making in Software Architecture", Hans van Vlieta, Antony Tang, 2015

Wir handeln oft intuitiv



Anekdoten-
getrieben

nach Bauchgefühl

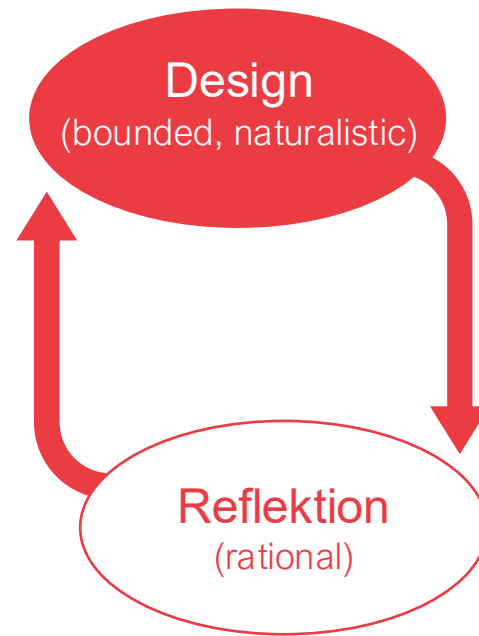
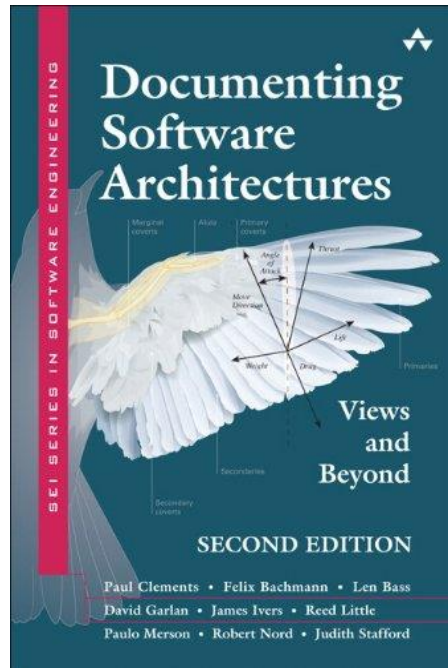
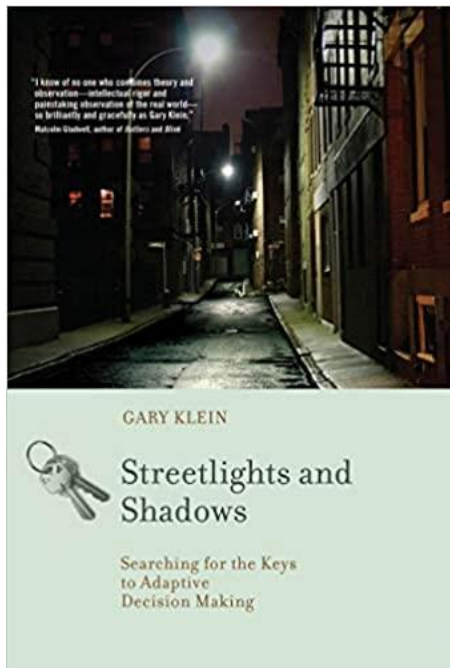
Konsens-
gesteuert

emotional

moralgetrieben

erfahrungsbasiert

Rationalität im nachhinein hilft





Review-Geschichte

Wo kommt das Thema her?
Warum weiß ich davon?





1998 1999

2000

2001 2002

ATAM – Architecture Tradeoff Analysis Method

„Konzeptioneller Fluss“ von ATAM

I'm familiar with architecture evaluation methods such as the technical Architecture Tradeoff Analysis Method (ATAM) and more business-oriented Cost Benefit Analysis Method (CBAM).

However, these methods are fairly large scale: they prescribe several brainstorming sessions, development of a host of scenarios describing tradeoffs, etc. While useful for internal projects or desktop applications that are

through the same 9 ATAM steps, but with less formality than what is described in the SEI's ATAM reference. Other ATAM reports that MITRE has participated in during the past have shown the ATAM to be a very "heavyweight" approach; the assessment of this project by necessity of the resource limitations and schedule demands had to be more of a "lightweight" assessment.

definitions from stakeholders and reviewing architectural evaluation methods. The lightweight factors are acquired by studying the five most commonly used lightweight methods and the Architecture-based Tradeoff Analysis Method (ATAM), the most well-known heavyweight method. Subsequently, the research addresses these features and

Non-Risks



2023 2024

2025

2026 2027

Bewertungsmethoden

SAAM

Software Architecture Analysis Method

ATAM

Architecture Tradeoff Analysis Method

CBAM

Cost-Benefit Analysis Method

TARA

Tiny Architecture Review Approach

PBAR

Pattern Based Architecture Review

ARID

Architecture Review for Intermediate Designs

DCAR

Decision Centric Architecture Review

DASE

Decision and Scenario based arch. evaluation

Pre-Mortem

Risiko-Brainstorming and Mitigation

LASR

Lightweight Approach for Software Reviews

03.

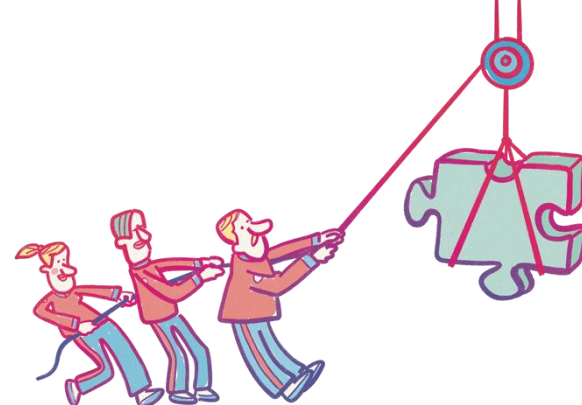
Kollaborative Methoden

Warum will man das?



Kollaborative Methoden

Fokus: Gemeinsames Verstehen



- **Alle Teilnehmenden** sind **aktiv**
- **Implizites Wissen** ist **sichtbar**
- Schafft **gemeinsames Verständnis**
- **Widersprüche** sind **direkt sichtbar**
- **Ownership** ist **stark**
- **Methoden** müssen **stärker ausgearbeitet** sein
- **Konflikte** sind **unmoderiert**



Besonders wertvoll bei komplexen, cross-funktionalen Fragestellungen

Kollaborative Methoden...



DDD & Domain Modeling

- Event Storming
- Domain Storytelling
- Example Mapping
- Context Mapping



Agil & Produktfokus

- Retrospektiven
- Backlog Refinement
- Three Amigos
- Value Stream Mapping



Engineering / Dev

- Pair / Mob Programming
- Brown Bag Sessions
- Threat Modeling
- Hackathon



UX & Discovery

- Design Thinking Workshops
- Design Sprints
- Customer-Journey Mapping
- Co-Design Workshops



Bewertungsmethoden

SAAM

Software Architecture Analysis Method

ATAM

Architecture Tradeoff Analysis Method

CBAM

Cost-Benefit Analysis Method

TARA

Tiny Architecture Review Approach

PBAR

Pattern Based Architecture Review

ARID

Architecture Review for Intermediate Designs

DCAR

Decision Centric Architecture Review

DASE

Decision and Scenario based arch. evaluation

Pre-Mortem

Risiko-Brainstorming and Mitigation

LASR

Lightweight Approach for Software Reviews

Stark moderierte Bewertungsmethoden

SAAM

Software Architecture Analysis Method

ATAM

Architecture Tradeoff Analysis Method

CBAM

Cost-Benefit Analysis Method

TARA

Tiny Architecture Review Approach

PBAR

Pattern Based Architecture Review

ARID

Architecture Review for Intermediate Designs

DCAR

Decision Centric Architecture Review

DASE

Decision and Scenario based arch. evaluation

Pre-Mortem

Risiko-Brainstorming and Mitigation

LASR

Lightweight Approach for Software Reviews

Individuelle Bewertungsmethoden

SAAM

Software Architecture Analysis Method

ATAM

Architecture Tradeoff Analysis Method

CBAM

Cost-Benefit Analysis Method

TARA

Tiny Architecture Review Approach

PBAR

Pattern Based Architecture Review

ARID

Architecture Review for Intermediate Designs

DCAR

Decision Centric Architecture Review

DASE

Decision and Scenario based arch. evaluation

Pre-Mortem

Risiko-Brainstorming and Mitigation

LASR

Lightweight Approach for Software Reviews



Kollaborative Bewertungsmethoden

SAAM

Software Architecture Analysis Method

ATAM

Architecture Tradeoff Analysis Method

CBAM

Cost-Benefit Analysis Method

TARA

Tiny Architecture Review Approach

PBAR

Pattern Based Architecture Review

ARID

Architecture Review for Intermediate Designs

DCAR

Decision Centric Architecture Review

DASE

Decision and Scenario based arch. evaluation

Pre-Mortem

Risiko-Brainstorming and Mitigation

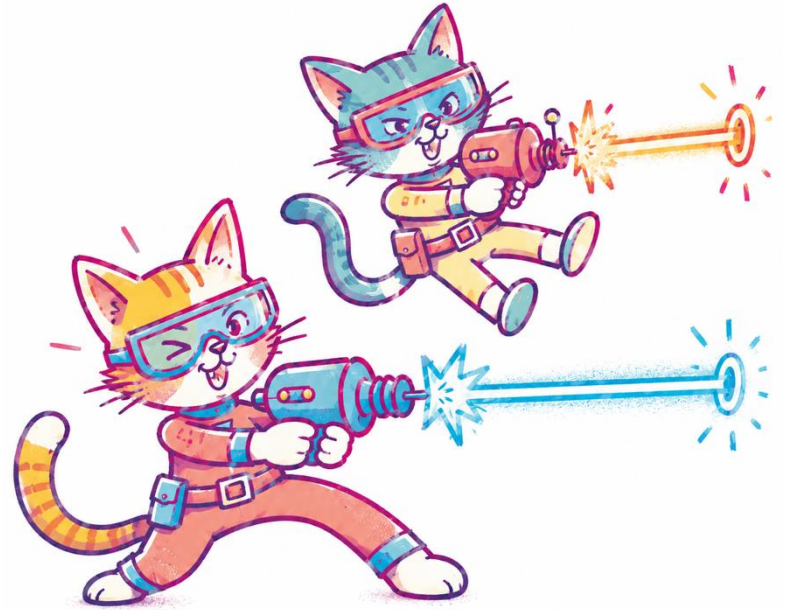
LASR

Lightweight Approach for Software Reviews

04.

LASR

Eine kollaborative, leichtgewichtige Review-Methode...

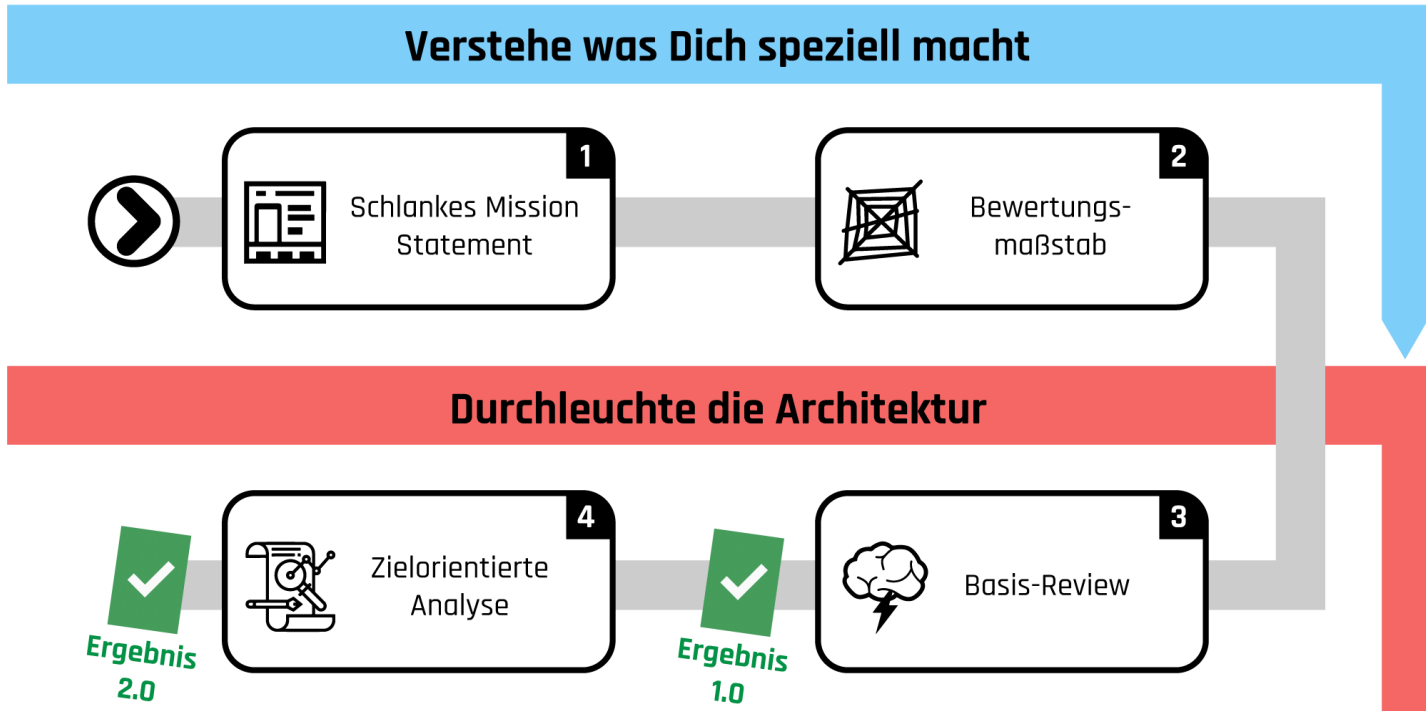


„LASR“



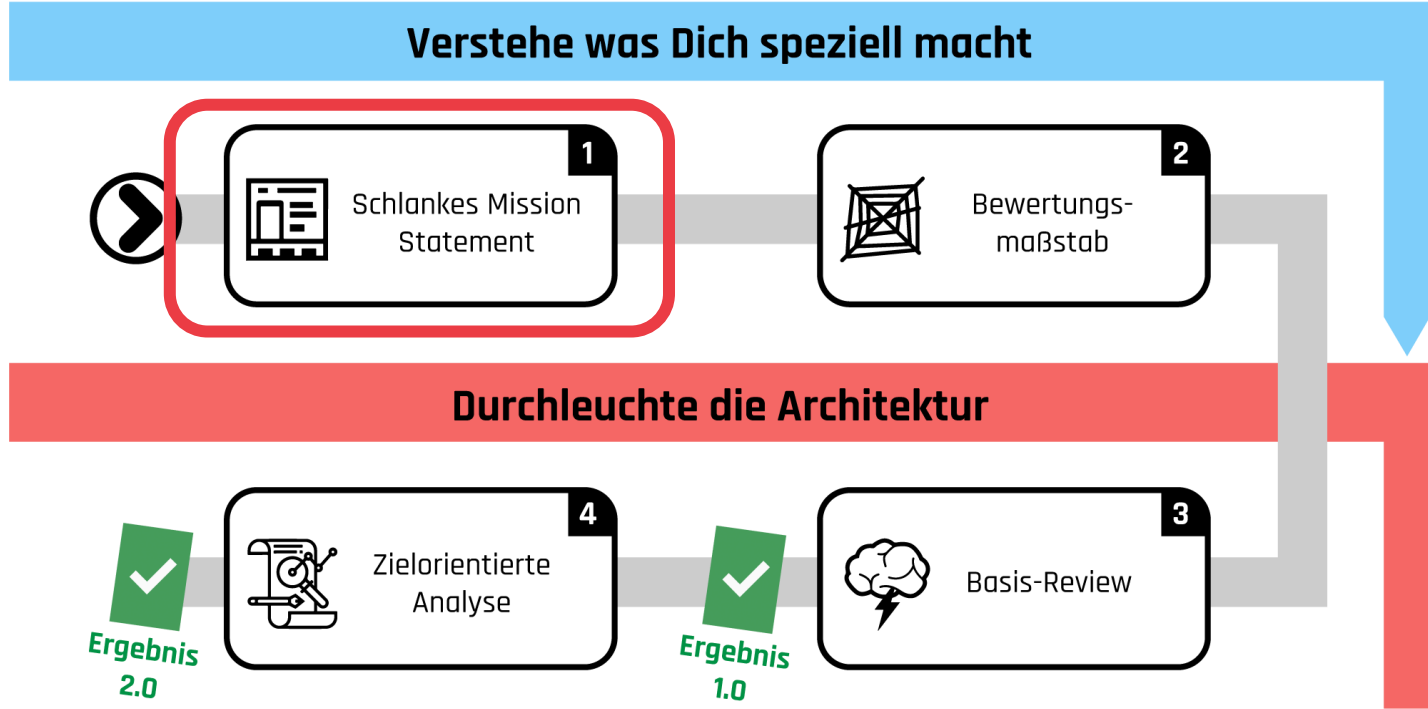


LASR Kernreview





Schritt 1: Schlankes Mission Statement



Verstehe was Dich speziell macht



Schlankes Mission Statement

1



Bewertungsmaßstab

2

Durchleuchte die Architektur



Ergebnis 2.0



Zielorientierte Analyse

4



Ergebnis 1.0



Basis-Review

3

Beispiel: Landing Page von Prometheus

prometheus.io

Prometheus

DOCS DOWNLOAD COMMUNITY SUPPORT & TRAINING BLOG

Search

From metrics to insight

Power your metrics and alerting with the leading open-source monitoring solution.

GET STARTED DOWNLOAD

Dimensional data

Prometheus implements a highly dimensional data model. Time series are identified by a metric name and a set of key-value pairs.

Powerful queries

PromQL allows slicing and dicing of collected time series data in order to generate ad-hoc graphs, tables, and alerts.

Great visualization

Prometheus has multiple modes for visualizing data: a built-in expression browser, Grafana integration, and a console template language.

Efficient storage

Prometheus stores time series in memory and on local disk in an efficient custom format. Scaling is achieved by functional sharding and federation.

Simple operation

Each server is independent for reliability, relying only on local storage. Written in Go, all binaries are statically

Precise alerting

Alerts are defined based on Prometheus's flexible PromQL and maintain dimensional information. An

Many client libraries

Client libraries allow easy instrumentation of services. Over ten languages are supported already and

Many integrations

Existing exporters allow bridging of third-party data into Prometheus. Examples: system statistics, as well as

Erarbeiten im Workshop

LASR Schritt 1: Schlankes Mission Statement

Was würde prominent auf der Landing Page der zu betrachtenden Softwarelösung stehen?



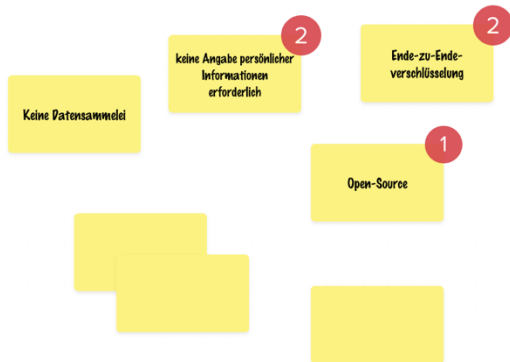
1 Brainstorming

Leitfragen:

Was sind zentrale Features oder Eigenschaften unserer Software?

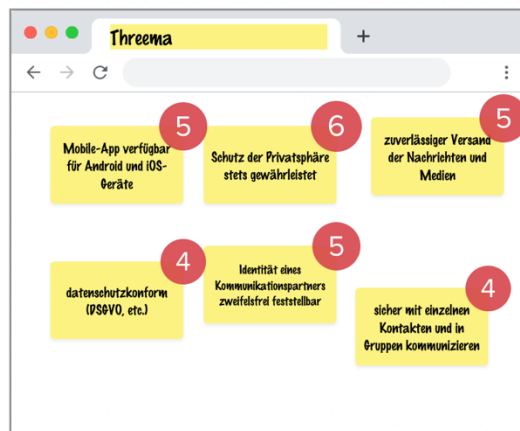
Was macht unsere Software besonders? Was besonders gut?

Welche Ansprüche haben wichtige Beteiligte an die Softwarelösung? ("Claims")



2 Clustern und Ausdünnen

Voting. Welche der gefundenen Punkte sollten Eurer Meinung nach auf jeden Fall auftauchen? Ziel: ca. 7 +/- 2.

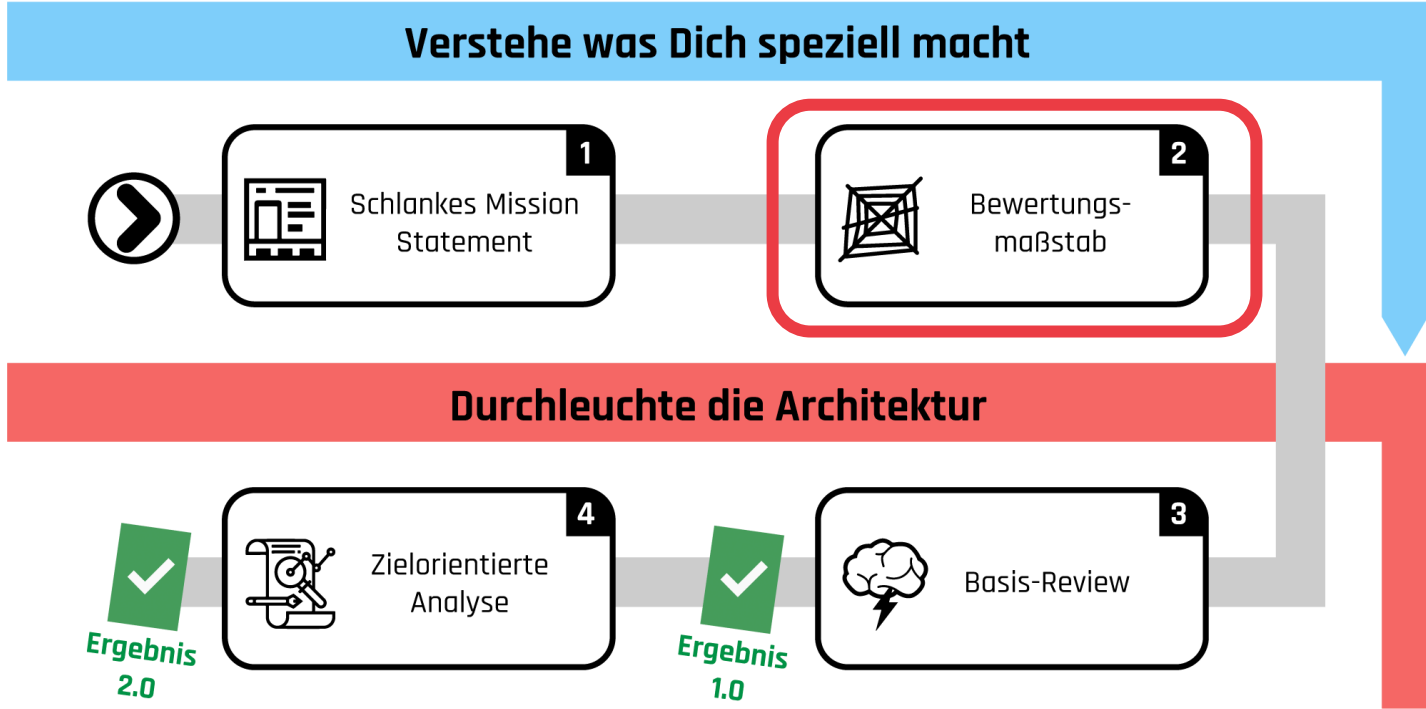


← Beispiel im digitalen Whiteboard (Mural)

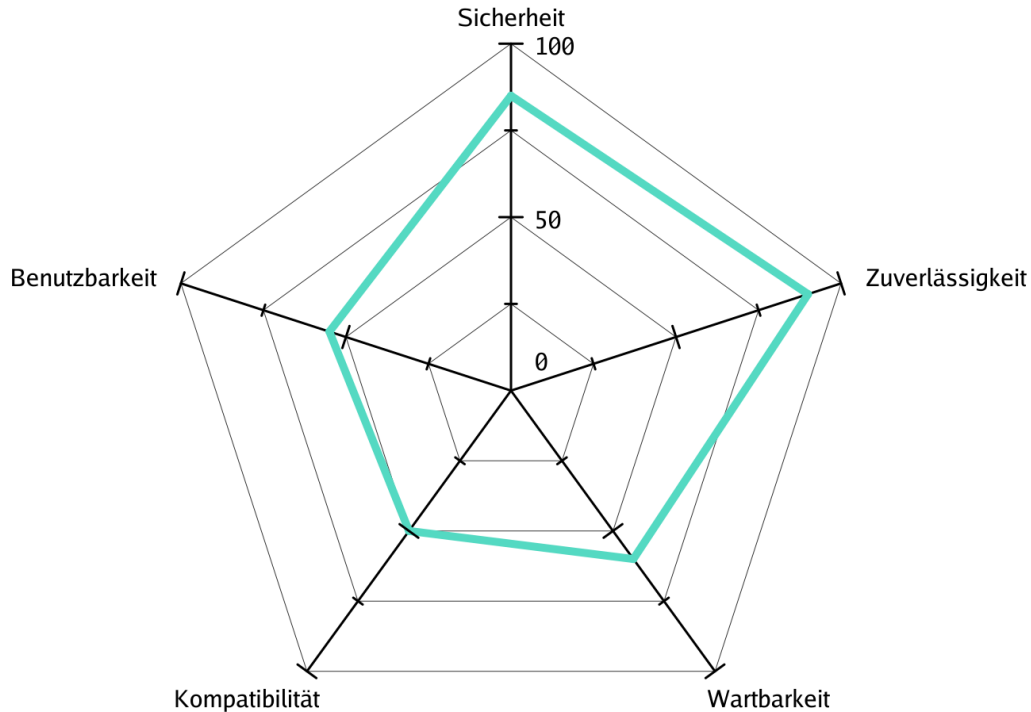
Quelle: S. Toth, S. Zörner:
„Software-Systeme
reviewen“, Leanpub 2023



Schritt 2: Bewertungsmaßstab



Bewertungsmaßstab in LASR



LASR-Ergebnisdiagramm

Die Top-3-5 Qualitätsziele bilden die Achsen des Diagramms.

Die Höhe der Ansprüche führt zu einer grünen Ziellinie.

Basis: Qualitätsmerkmale

Begriffe
nach
ISO 25010



Funktionale Eignung (Functional Suitability)

Sind die berechneten Ergebnisse genau genug / exakt, ist die Funktionalität angemessen? ...



Zuverlässigkeit (Reliability)

Ist das System verfügbar, tolerant gegenüber Fehlern, nach Abstürzen schnell wieder hergestellt? ...



Benutzbarkeit (Usability)

Ist die Software intuitiv zu bedienen, leicht zu erlernen, attraktiv? ...



Effizienz (Performance)

Antwortet die Software schnell, hat sie einen hohen Durchsatz, einen geringen Ressourcenverbrauch? ...



Sicherheit (Security)

Ist das System sicher vor Angriffen? Sind Daten und Funktion vor unberechtigtem Zugriff geschützt? ...



Portabilität (Portability)

Ist die Software leicht auf andere Zielumgebungen (z.B. anderes OS) übertragbar? ...



Kompatibilität (Compatibility)

Ist die Software konform zu Standards, arbeitet sie gut mit anderen zusammen? ...



Wartbarkeit (Maintainability)

Ist die Software leicht zu ändern, erweitern, testen, verstehen? Lassen sich Teile wiederverwenden? ...

14 Qualitätsmerkmale als Zielsetzung

Funktionale Eignung



Sind die berechneten Ergebnisse genau genug / exakt, ist die Funktionalität angemessen?

Angemessenheit

Korrektheit

Vollständigkeit

Zuverlässigkeit



Ist das System verfügbar, tolerant gegenüber Fehlern, nach Abstürzen schnell wieder hergestellt?

Verfügbarkeit

Wiederherstellbarkeit

Fehlertoleranz

Performanz



Antwortet die Software schnell, hat sie einen hohen Durchsatz, einen geringen Ressourcenverbrauch?

Zeitverhalten

Verbrauchsverhalten

Kapazität

Benutzbarkeit



Ist die Software intuitiv zu bedienen, wiedererkennbar, leicht zu erlernen, attraktiv?

Bedienbarkeit

Erlernbarkeit

Barrierefreiheit

Sicherheit



Ist das System sicher vor Angriffen? Sind Daten und Funktion vor unberechtigtem Zugriff geschützt?

Vertraulichkeit

Authentizität

Datenintegrität

Betriebbarkeit



Lässt sich die Software gut betreuen? Sind Produktionsprobleme leicht auffindbar und behebbar?

Beobachtbarkeit

Aktualisierbarkeit

Reaktionseffizienz

Skalierbarkeit



Kann die Software auf Lastschwankungen und Wachstum etwa in den Mengengeräten angemessen reagieren?

Deployment-Flexibilität

Elastizität

Wachstumseffizienz

Nachhaltigkeit




Ist die Software umweltverträglich? Lässt sie sich ressourcenschonend nutzen?

Energieeffizienz

Emissionsarmut

Langlebigkeit

Auditierbarkeit



Ist die Software aus einer juristischen, finanziellen oder Sicherheitsperspektive gut zu überprüfen?

Nachvollziehbarkeit

Nachweisbarkeit

Zurechenbarkeit

Safety



Sind Personen, Tiere, Sachen oder Umwelt vor Schäden durch die Software geschützt?

Betriebsicherheit

Schützbarkeit

Regulatorische Compliance

Kosteneffizienz



Ist der Betrieb der Software wirtschaftlich optimiert und lässt sich ihr Einsatz gut planen?

Sparsamkeit

Schützbarkeit

Kostentransparenz

Wartbarkeit



Ist die Software leicht zu ändern, erweitern, testen, verstehen? Lassen sich Teile wiederverwenden?

Analysierbarkeit

Änderbarkeit

Erweiterbarkeit

Kompatibilität



Ist die Software konform zu Standards, arbeitet sie gut mit anderen zusammen?

Koexistenz

Interoperabilität

Versionierungsfähigkeit

Portierbarkeit



Ist die Software leicht auf andere Zielumgebungen übertragbar?

Übertragbarkeit

Installierbarkeit

Austauschbarkeit

Top-5-Challenger (aus 14 Karten die 5 wichtigsten wählen)



Start Top-5



Kandidaten

Top-5-Challenger (after 2 rounds)

Maintainability

Is the software easily modified and extended? Is it well tested and understood? Are parts all reusable?

Adaptability
 Modifiability
 Extensibility

Scalability

Can the software meet organically to growing or shrinking workloads? Can it adjust its capacity to handle variability?

Flexibility of Deployment
 Elasticity
 Growth Efficiency

Cost Efficiency

Is the operation of the system economically optimized? Is it easy to budget for on-peak, change of scaling?

Autonomy
 Ability to Estimate
 Cost Transparency

Functional Suitability

Are calculated results accurate enough, are measurements exact, is the functionality optimized?

Appropriateness
 Correctness
 Completeness

Performance Efficiency

Does the software perform within given time and throughput parameters? Does it make efficient use of resources?

Time Behaviour
 Resource Utilization
 Capacity

Sustainability

Is the development and operation of the software environmental friendly? Is resource consumption optimized?

Energy Efficiency
 Low Emissions
 Longevity

Operability

Is the software easily operated and supported? Are production problems easy to find and fix?

Observability
 Updatability
 Response Efficiency

Top-5

Auditability

Is it easy to review or verify the software from a legal, financial or security perspective?

Traceability
 Verifiability
 Attestability

Compatibility

Is the software compatible with hardware? Is it able to exchange information or data transparently with other systems?

Co existence
 Interoperability
 Interworking Capability

Usability

Is it intuitive to use the software? Is it recognizable, easy to learn, attractive? Is it hard to use the software as a user?

Discoverability
 Learnability
 Inclusivity

Portability

Is the software easily transferred to different hardware environments? Is it easily adapted for different contexts?

Adaptability
 Installability
 Replaceability

Safety

Are errors, crashes, corruption or the environment protected from damage by the system?

Operational Safety
 Verifiability
 Regulatory Compliance

Security

Is the system secured against attacks? Are data and functionality protected against unauthorized access?

Confidentiality
 Authenticity
 Integrity of Data or State

Reliability

Is the system available, fault tolerant, and quickly restored after crashes or outages?

Availability
 Recoverability
 Fault Tolerance

Candidates

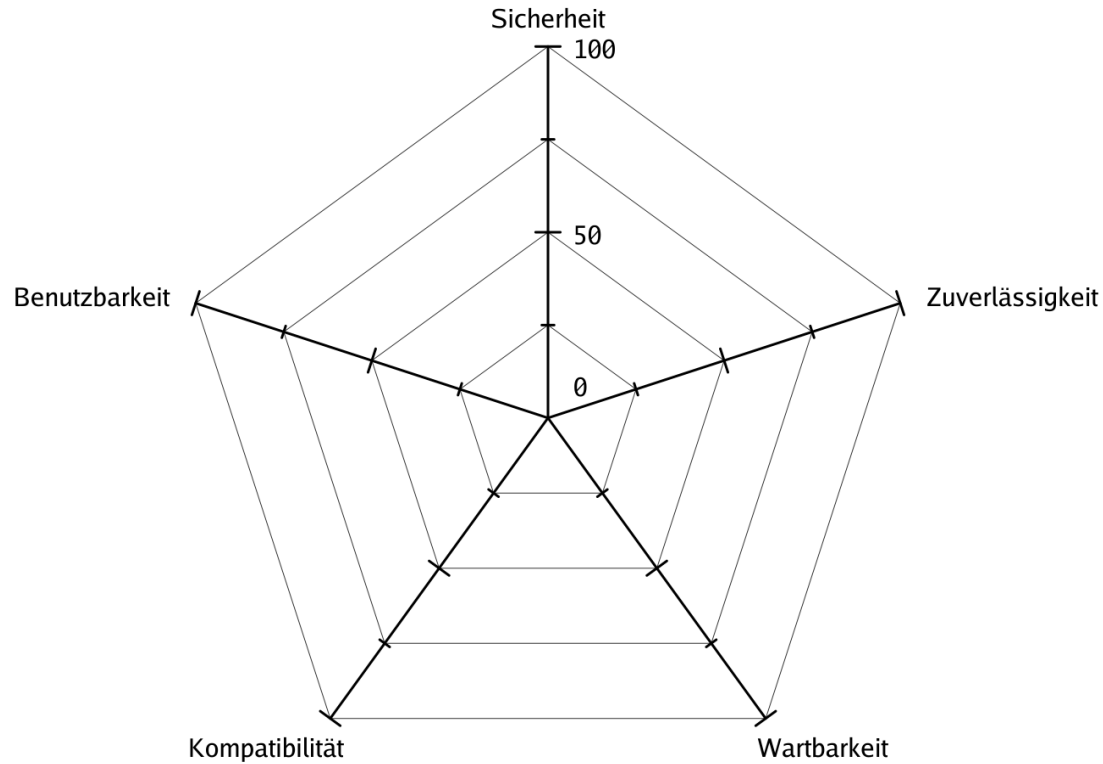


Impression aus Workshops





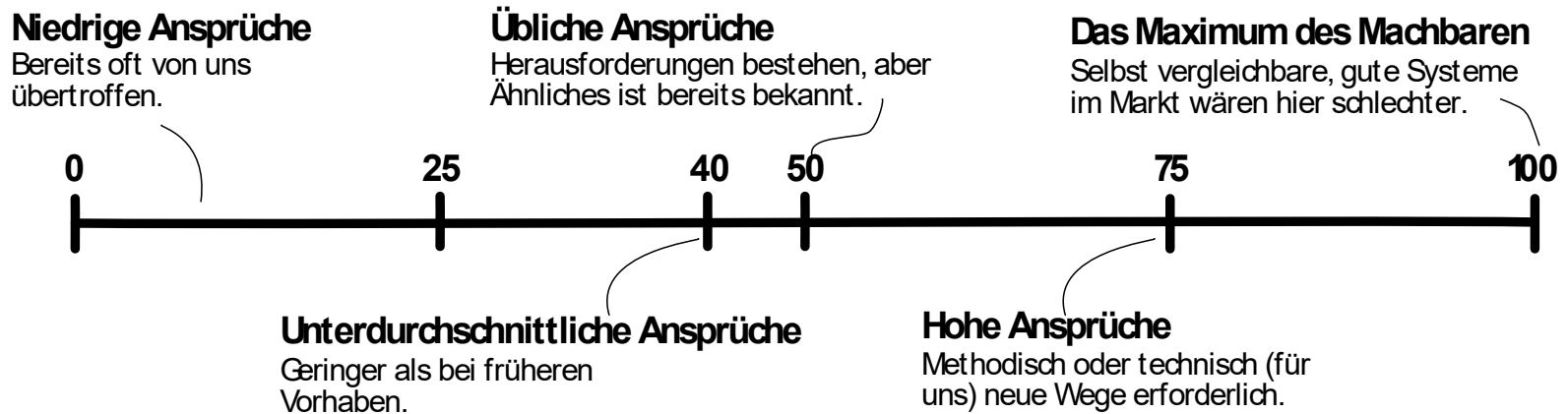
Die Top 3-5 Qualitätsziele identifiziert (Beispiel)





Zielwerte bestimmen

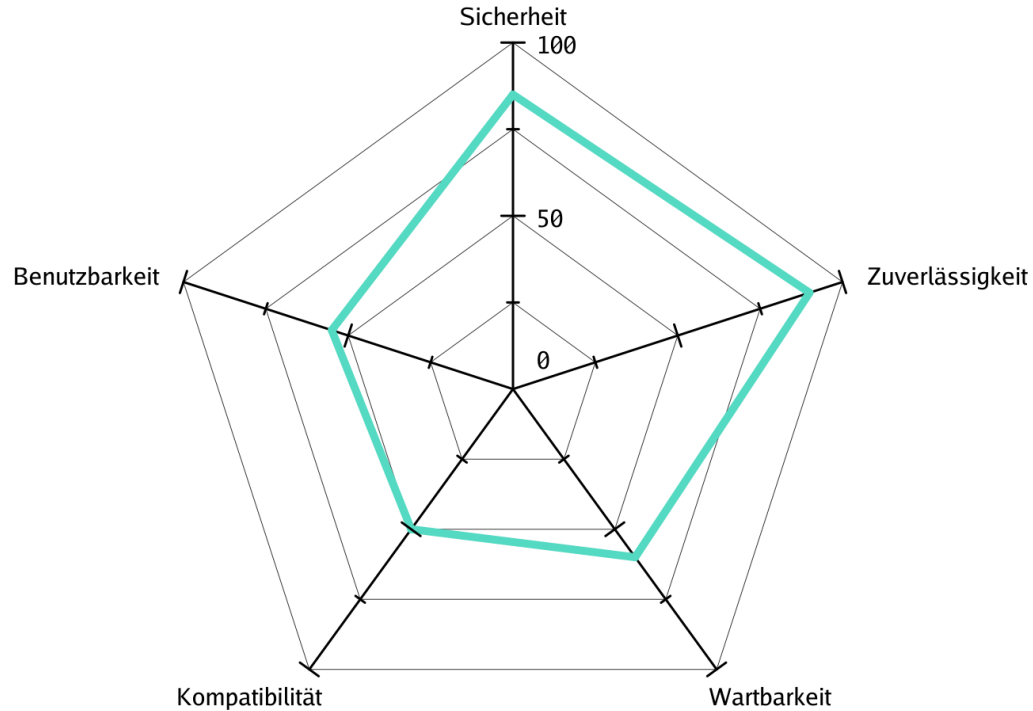
Der Zielwert zwischen 0 und 100 beschreibt jeweils, wie hoch die Erwartungen in dem Bereich sind. Im Vergleich zu anderen Zielen der Top-3-5 und anderen Vorhaben aus dem Kontext des Unternehmens / der Organisation.



Hinweis: Es handelt sich hier um eine Einschätzung, nichts Messbares.

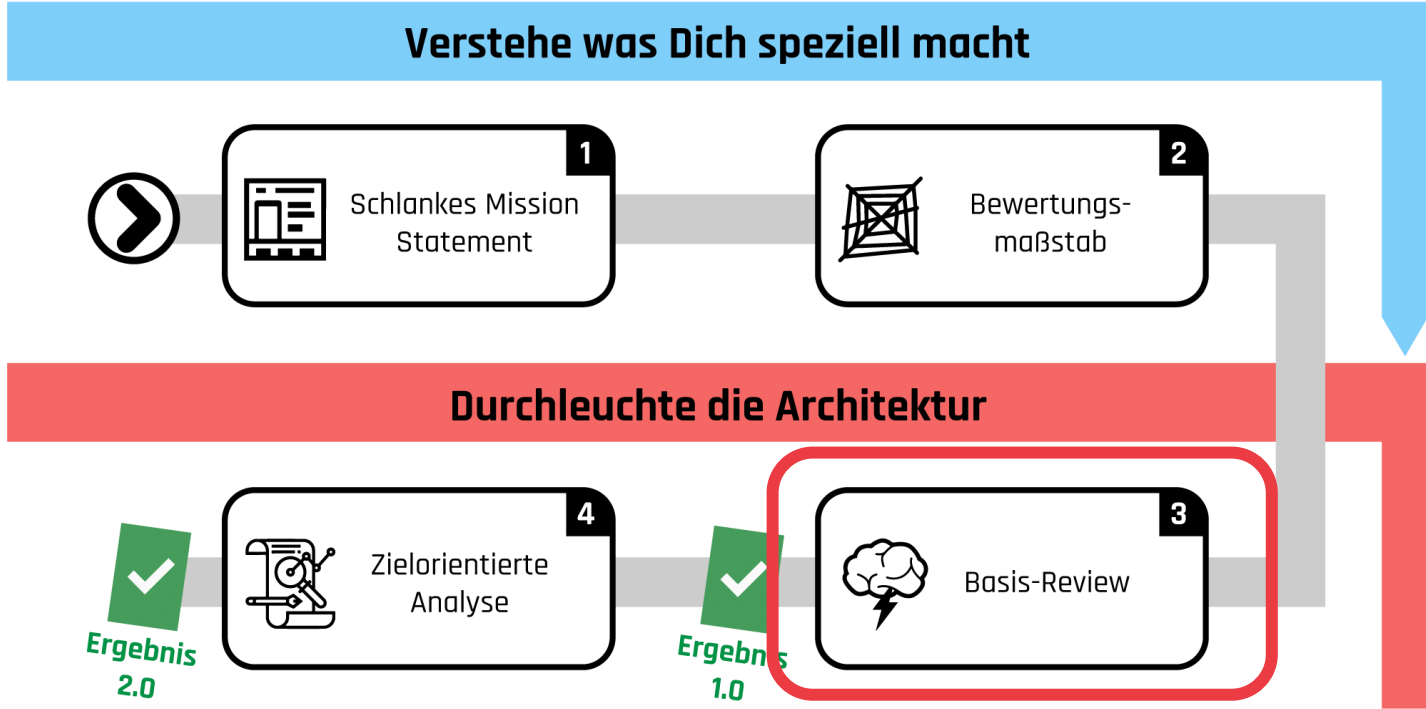


Bewertungsmaßstab eingezeichnet (Beispiel)





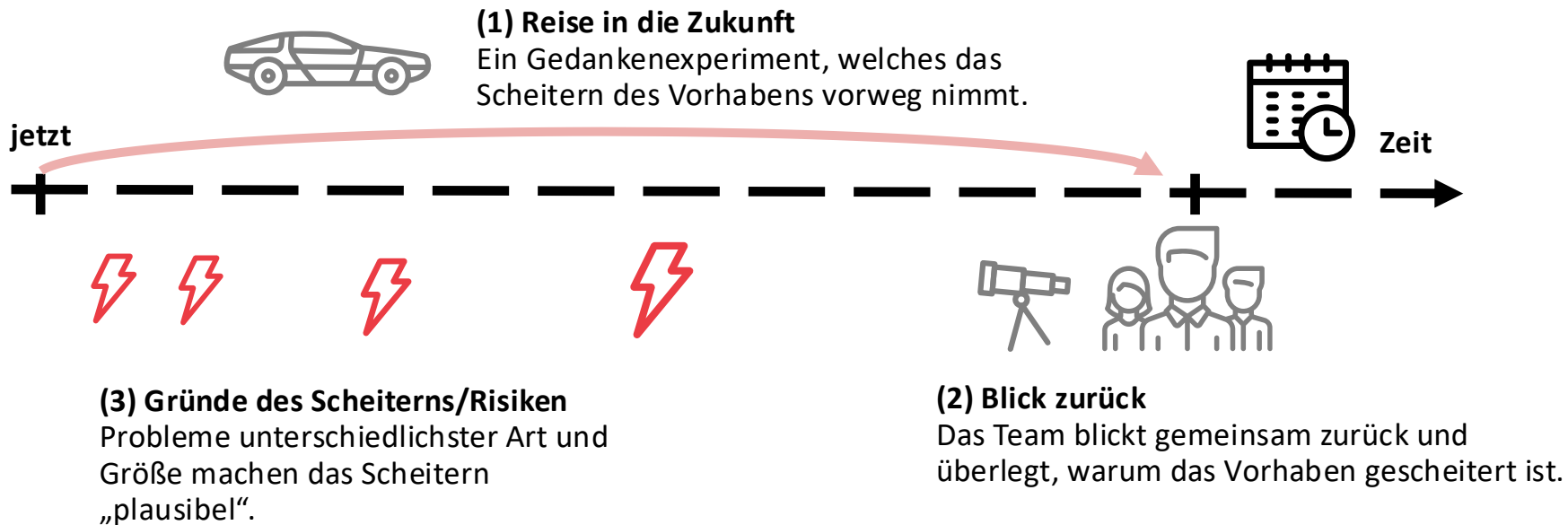
Schritt 3: Basis-Review



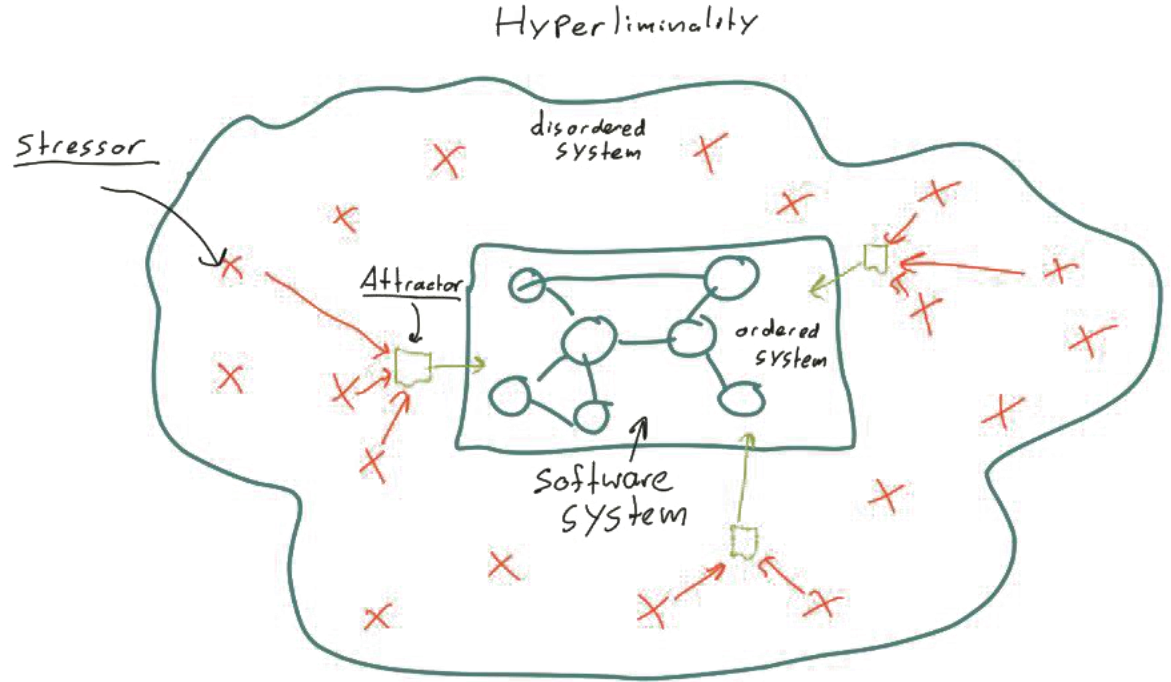
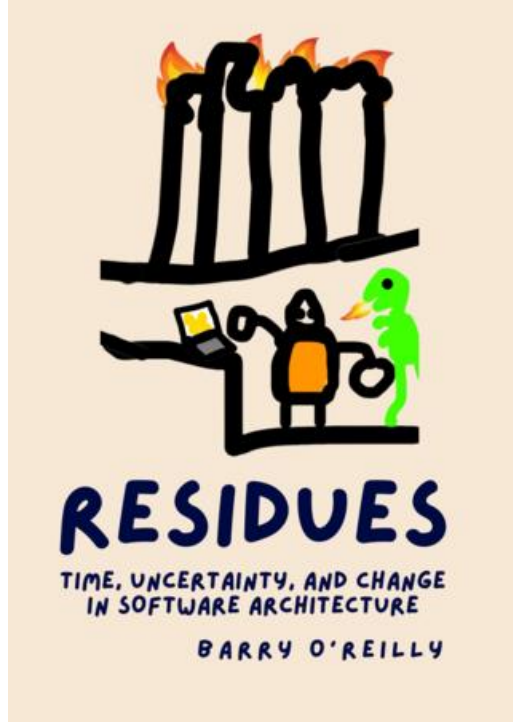
Grundidee: Pre Mortem



Was ist ein Pre-Mortem?



Zufall oder Wissenschaft?



Typische Risikothemen als Ideengeber



Zu hohe Komplexität der Lösung

Hat die Domäne eine sehr hohe Komplexität? Gibt es unüberlegte, schnelle Lösungen oder fehlende Abstraktionen? Komplexität gefährdet den Überblick bzw. Wartbarkeit, Korrektheit, Sicherheit, ...

 Softwarelösung 1.1



Unpassende Lösungs-Strukturierung

Folgt die Softwarestruktur der Domäne? Sind andere technische oder organisatorische Einflüsse (Conway...) gut aufgegriffen? Falls nicht könnte Wartbarkeit, Zuverlässigkeit etc. leiden.

 Softwarelösung 1.2



Inadäquates Daten-Handling

Ist die Ablage, der Transport und das Mapping von Daten konzeptionell und technisch passend gelöst? Sind Daten ausreichend schnell und rechtssicher, im richtigen Format an den richtigen Stellen?

 Softwarelösung 1.3



Problematische Konzepte & Technologien

Passen die eingesetzten Muster und Konzepte zu den Zielen? Sind sie konsistent angewendet? Sind die verwendeten Technologien und Frameworks passend und etabliert?

 Softwarelösung 1.4



Softwarelösung

1

- Unpassende Technologien
- Komplexe Lösungen
- Unausgereifte Fremdlösungen
- Behindernde Frameworks / ...
- Strukturprobleme

Brainstorming-Unterstützung: Das LASR-Kartenset hält insgesamt 32 konkrete Risikokarten als Ideengeber parat, 4 in jeder der 8 Kategorien.

Die 8 Risiko-Kategorien von LASR

Softwarelösung 1

- Unpassende Technologien
- Komplexe Lösungen
- Unausgereifte Fremdlösungen
- Behindernde Frameworks / Libraries
- Strukturprobleme
- ...

Kompetenz und Erfahrung 2

- Fehlendes oder isoliertes Wissen
- Zu kleine Lernfenster
- Fehlendes Prozessverständnis
- Tool-Probleme
- Schwere Schätzbarkeit
- ...

Zielsetzungen und Erwartungen 3

- Unrealistische Ziele
- Überzogene Erwartungen
- Knappe Deadlines
- Instabile Anforderungen
- Fehlender Kundenkontakt
- ...

Fremdsysteme und Plattformen 4

- Instabile Plattformen
- Fehleranfällige Fremdsysteme
- Unpassende Buy/ Take Wahl
- Lizenzschwierigkeiten
- Vendor Lock-In
- Integrationsprobleme
- ...

Altsysteme und Altlasten 5

- Legacy-Behinderungen
- Innovationsstau
- Zerbrechliche Lösungen
- Fehlende Tests
- Wenig Dokumentation
- Fehlendes Verständnis
- ...

Organisation und Prozesse 6

- Geschwollene Prozesse
- Behindernde Rollenmodelle
- Organisationsgrenzen
- Politik
- Engen Standards
- Isolierte Entscheidungskompetenzen
- ...

Betrieb und Deployment 7

- Blockierende Prozesse
- Geringe Automatisierung
- Wenig Feedback
- Fehlende Betriebs- / Ausfallkonzepte
- Tool-Probleme
- ...

Weiche Faktoren 8

- Uneinigkeiten
- Konflikte
- Fehlende Disziplin
- Kommunikationsbarrieren
- Rollenhermatiken
- Unpassende Kultur
- ...

20 min in der Zukunft...

SAF-Vortrag ist gescheitert... Warum?



Zu hohe Ziele oder Erwartungen

Sind Qualitätsziele (z.B. Performanz) zu ambitioniert? Stehen hohe Einzelziele guter Gesamtqualität im Weg? Werden Randbedingungen verletzt oder unnötige technische Risiken eingegangen?



Zielsetzungen & Erwartungen

3.1



Falsche / fehlende Tool-Unterstützung

Behindern Werkzeuge bei Änderungen, Betrieb oder Kommunikation? Fehlen wichtige Tools oder Prozesse für hohe Development Maturity (CI/CD)? Sind repetitive Aufgaben manuell abgebildet?



Kompetenz & Erfahrung

2.3



Negative Seiteneffekte anderer Themen

Gibt es potentiell Behinderungen durch Nachbarprojekte oder fremde Organisationseinheiten? Gibt es konkurrierende Tätigkeiten, die Budget, Verfügbarkeiten oder Technologiewahl beeinflussen?

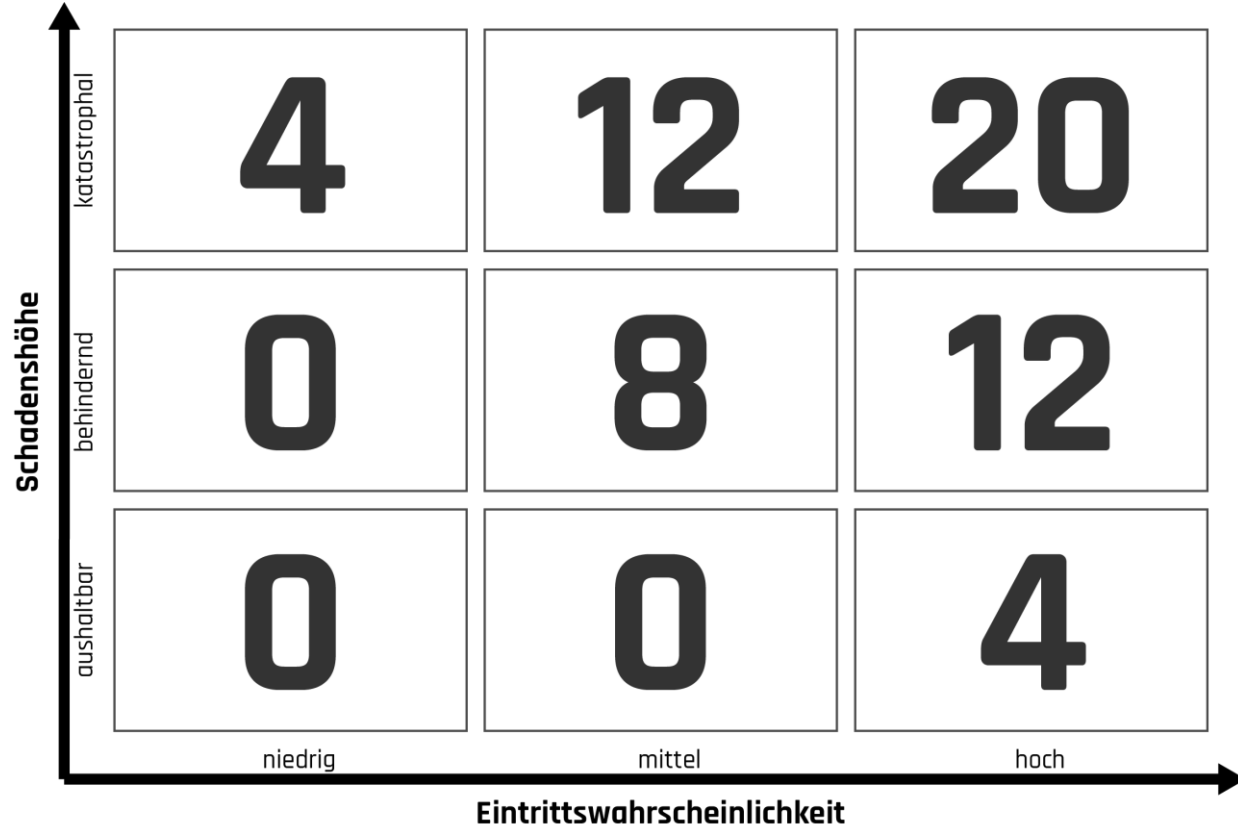
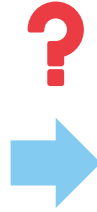


Fremdsysteme & Plattformen

4.1



Risikokarten einordnen





Fehlendes Lösungsverständnis

Fehlen Wissen, Ansprechpartner und/oder Dokumentation für Teile des Systems?
Sitzen Ansprechpartner an Bottenrücken?
Sind zentrale Lösungen (w. wenig nachvollziehbar oder "historisch gewachsen")

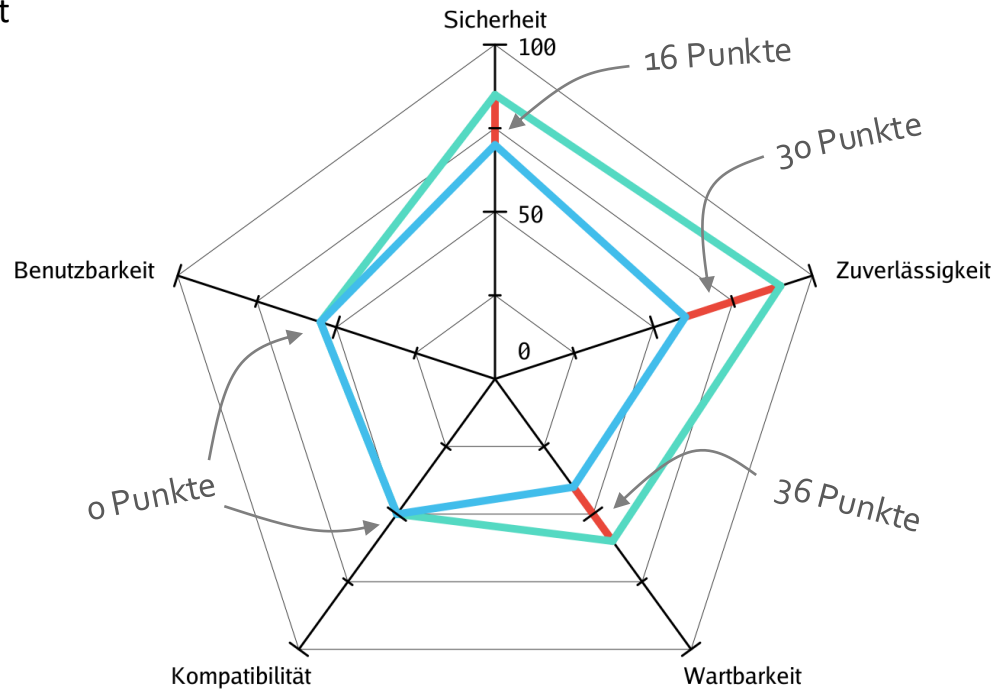
Altsysteme und Altlasten

53



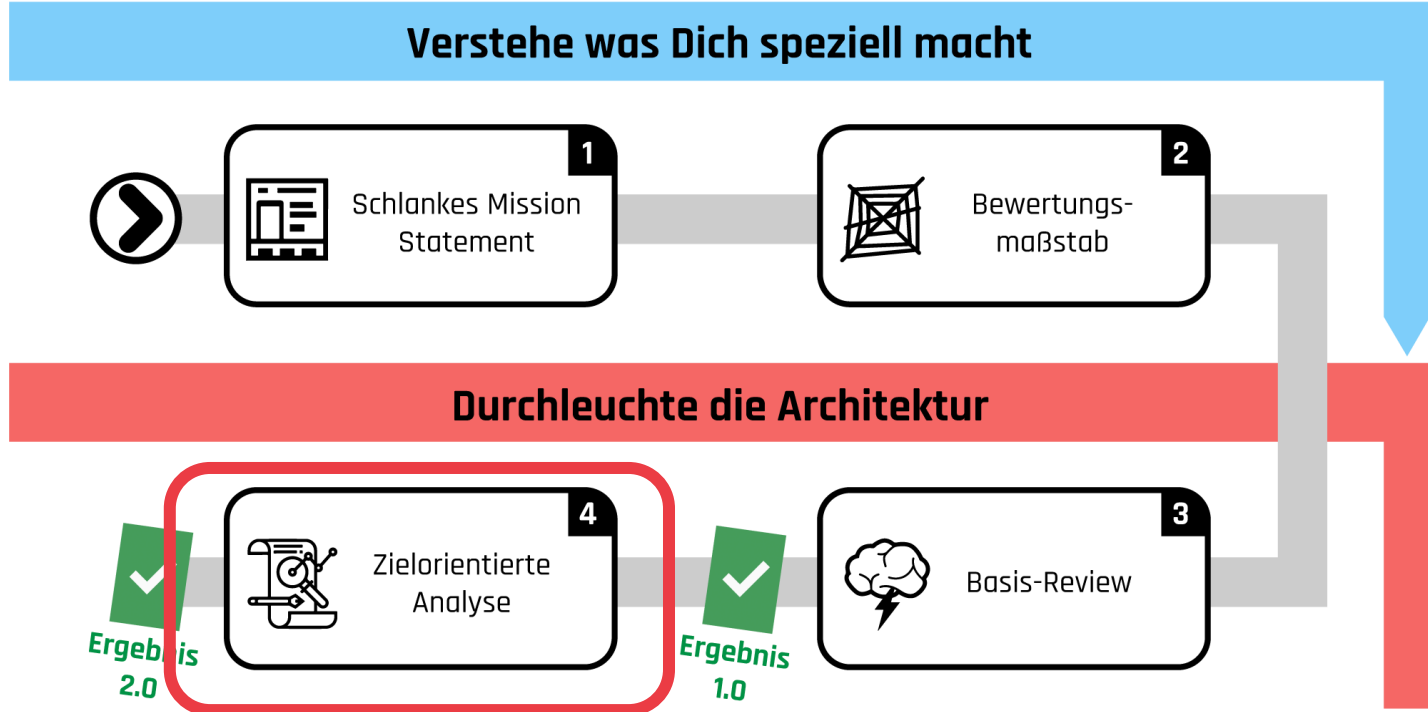
Lücken einzeichnen

Gesamtpunktezahl der dem Qualitätsziel zugeordneten Karten = Abweichung von grüner Linie in Prozent ($>100 = 100$)





Schritt 4: Zielorientierte Analyse



Verstehe was Dich speziell macht



1
 Schlankes Mission Statement

2
 Bewertungsmaßstab

Durchleuchte die Architektur



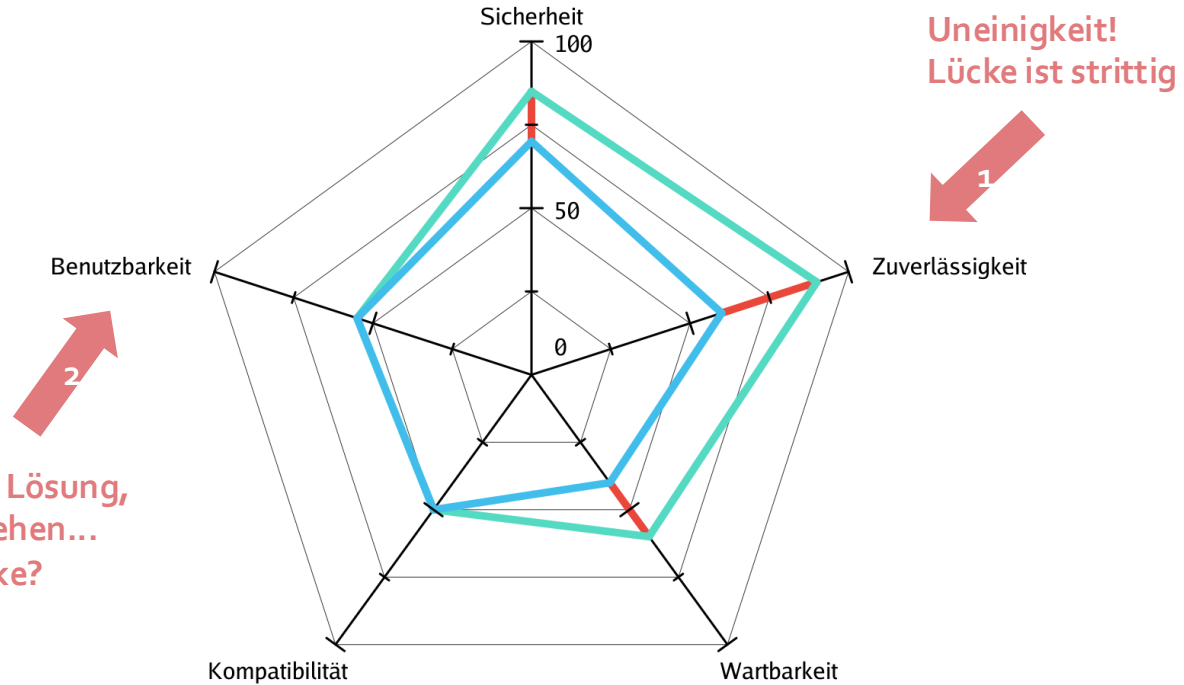
4
 Zielorientierte Analyse



3
 Basis-Review



Fokus für Zielorientierte Analyse - Bsp



Uneinigkeit!
Lücke ist strittig



Keine Stärken in der Lösung,
Kein explizites Vorgehen...
Trotzdem keine Lücke?



Template

Vorlage, um die Ergebnisse der Analyse zu sammeln, zu verdichten und TODOs abzuleiten.

Beispiel →

Zielachse

Lücke (Schritt 3):

36

Lücke geschätzt:

20

Zuverlässigkeit ISO-Std



ist das System verfügbar, tolerant gegenüber Fehlern, nach Abstürzen schnell wieder hergestellt?

Verfügbarkeit _____

Wiederherstellbarkeit _____

Fehlertoleranz _____

Qualitätsaussagen

Welche Aussagen illustrieren die Zielachse?

H Risiko: H/M/L

! Mehrstündigen Netzverlust auf 50-60selle -> Notlauf trotzdem gewährleistet

L Risiko: H/M/L

! Die Verbindung zum zentralen Server ist unterbrochen. Die Anlage sollte maximal 30 min unbenutzbar

L Risiko: H/M/L

Es kommt zu einem Daten-berührungspunkt. Es ist keine Nutzer-Interaktion erforderlich (Status anzeigen, protokollieren und Bergr.).

! Zentrale QAs - Qualitätsaussagen

H Technisches Risiko dass QA erreicht wird: High / Medium / Low

Konkrete Risiken

Welche Risiken sind der Zielachse zugeordnet?

2C Materialdaten wären bei Netzverlust abgeschoben. Nachgelagerte Probleme? keine QA betroffen **!** leichte Mitigation

2B Verbindungswächler bei Problemen manuell nicht automatisch keine QA betroffen **!** leichte Mitigation

1C SLAs zu Netzwerkverbindung ungeklärt keine QA betroffen **!** leichte Mitigation

3B Erkennung von Ausfall und Unsache ist manuell keine QA betroffen **!** leichte Mitigation

1C Risiko-Einschätzung

	1C	2C	3C
Betroffenheit	1B	2B	3B
	1A	2A	3A
	Eintrittswahrscheinlichkeit		

Stärken der Lösung

Welche Lösungen bringen uns Zielen näher?
Welche Aspekte erleichtern Umgang mit Risiken?

Message-basierte Kommunikation (async, retry-fähig)

Produktion ohne aktive Verbindung möglich (Arbeitsformate bei liegt one)

Versorgung der V-DG auch asynchron - auch asynchron - Daten gehen nicht verloren

Verarbeitungs-komponente kann Jobs ohne Inhalt überbrücken

Schwächen der Lösung

Welche Lösungsaspekte wirken problematisch?
Welche Aspekte verschulden Risiken?

Fehlererkennung über Videos kostet viel Bandbreite

Noch keine Erkennung von Verbindungs-problemen

Datenintensive Verbindung über VPN

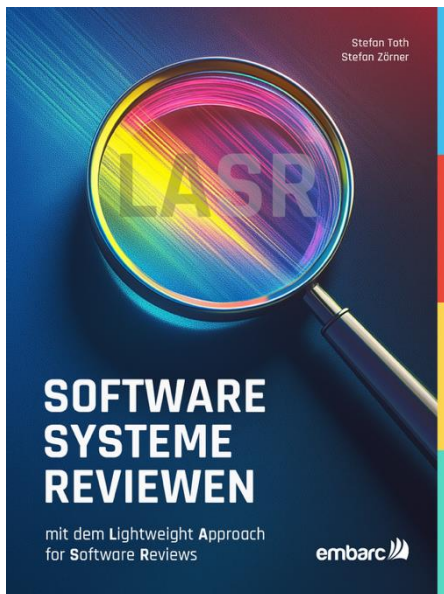
05.

Abschluss

Weitere Informationen...



Buchtipp zum Thema



Software-Systeme reviewen mit dem Lightweight Approach for Software Reviews - LASR

Autoren: Stefan Toth, Stefan Zörner
Verlag: Leanpub, September 2023
Sprache: Deutsch, EPUB, PDF



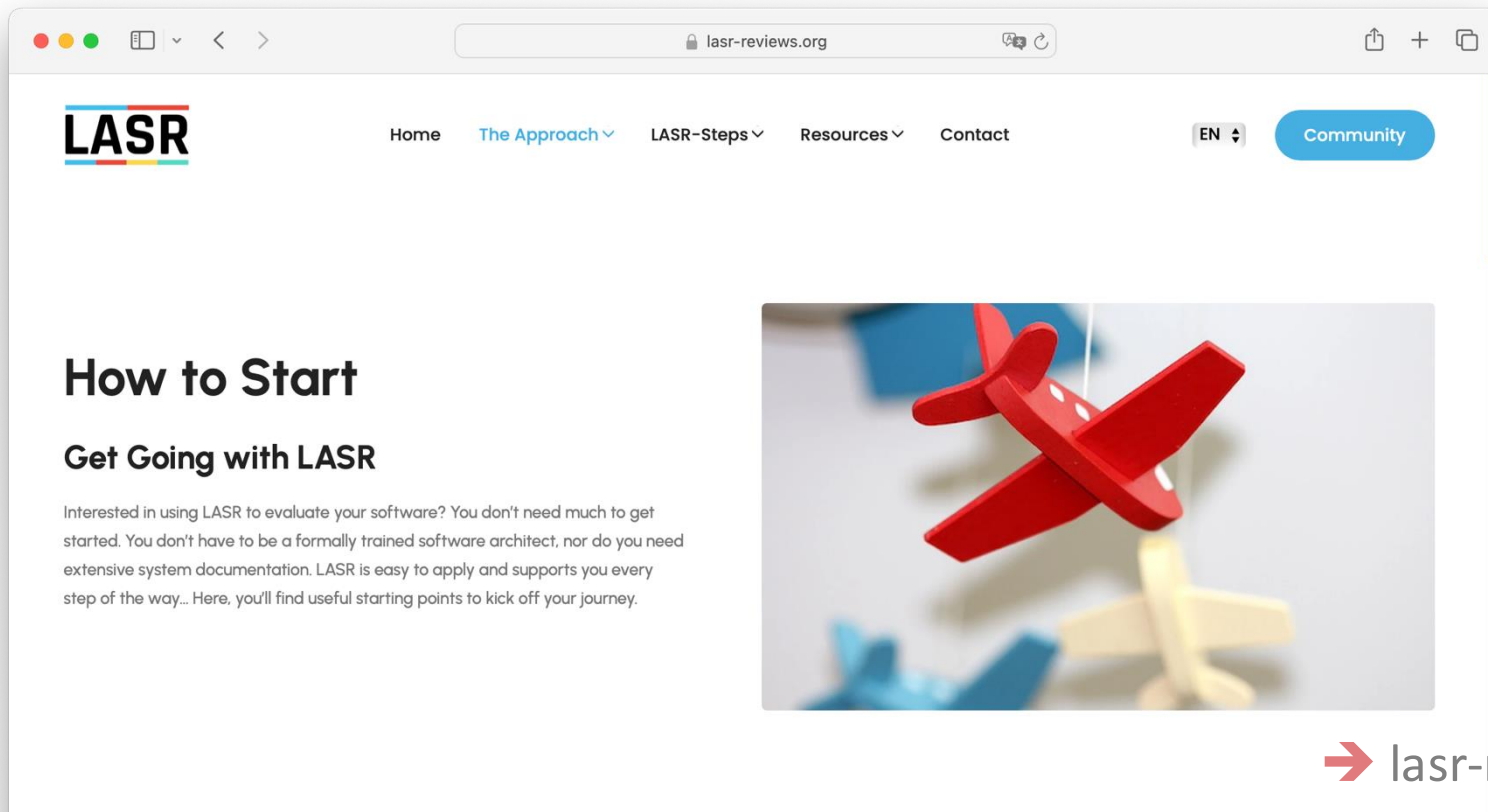
→ leanpub.com/software-systeme-reviewen/

Risiko-Kartenset bestellen

→ <https://www.embarc.de/lasr-kartenset-bestellen/>



„Offizielle“ LASR-Webseite (EN + DE)




The screenshot shows the homepage of the LASR website. The browser address bar displays "lasr-reviews.org". The navigation menu includes "Home", "The Approach", "LASR-Steps", "Resources", and "Contact". A language selector shows "EN" and a "Community" button is visible. The main content area features the heading "How to Start" and "Get Going with LASR". Below this, a paragraph explains that LASR is easy to apply and supports users every step of the way. To the right of the text is a photograph of a red toy airplane hanging from a string, with other blurred toy airplanes in the background.

Home The Approach LASR-Steps Resources Contact EN Community

How to Start

Get Going with LASR

Interested in using LASR to evaluate your software? You don't need much to get started. You don't have to be a formally trained software architect, nor do you need extensive system documentation. LASR is easy to apply and supports you every step of the way... Here, you'll find useful starting points to kick off your journey.





ARCEVAL – Software-Architekturbewertung

 **SOCREATORY**

▶ TRAININGS

▶ THEMEN

TRAINER:INNEN

▶ LEISTUNGEN

NEWS


KONTAKT

DE | EN

Home > Trainings > Architekturbewertung

Architekturbewertung

Training iSAQB® CPSA®-Advanced ARCEVAL – 2 Tage

Technik  Methodik  Kommunikation 

Vor-Ort-Termine

ARCEVAL (Therme Erding) - Stefan Toth - Deutsch - Termingarantie

30. Juni – 1. Juli 2026

Jetzt buchen

ARCEVAL (Hamburg) - Stefan Toth - Deutsch - 2 Tickets verfügbar

4.–5. November 2026

Jetzt buchen

Event-Ticketing-Software von pretix

<https://www.socreatory.com/de/trainings/arceval>

Feedback & Fragen?

Ich freue mich auf Fragen,
Diskussionen, Ausprobieren!

