Stefan Zörner

Softwarearchitektur in Eigenregie bewerten. Mit LASR in 2-3 Stunden zu ersten Review-Ergebnissen

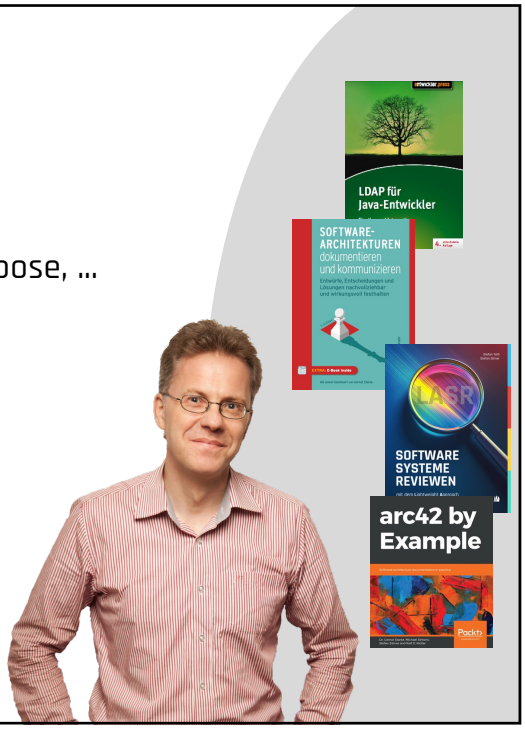
Vorträge bei der JUG CH
Bern, 15. September 2026
Zürich, 16.September 2026

Stefan Zörner

- Softwarearchitekt bei embarc in Hamburg
- Vorher Bayer AG, Mummert + Partner, IBM, oose, ...

Schwerpunkte

- Methodische Softwarearchitektur (Entwurf, Bewertung, Dokumentation)
- Architektur-Reviews



Abstract

Softwarearchitektur in Eigenregie bewerten.
Mit LASR in 2-3 Stunden zu ersten Review-Ergebnissen

Mit Architekturbewertungen ist es möglich, Schwächen und Potenziale von Softwarelösungen herauszuarbeiten, Entscheidungen abzusichern und Verbesserungsmaßnahmen zu bewerten. Klassische Analyseansätze aus diesem Umfeld wie ATAM sind fundiert, kommen aber gerade in beweglichen Softwarevorhaben etwas schwergewichtig, mitunter fast zeremoniell daher.

In diesem interaktiven Vortrag lernt ihr daher mit LASR (Lightweight Approach for Software Reviews) eine leichtgewichtige Herangehensweise kennen. Ihr könnt diese mit eurem Team unmittelbar anwenden, euer Softwaresystem beleuchten und zügig zu ersten Erkenntnissen kommen.

Wir greifen auf die Essenzen etablierter Bewertungsmethoden zurück und erarbeiten uns einen roten Faden durch ein Review, inkl. möglicher Vertiefungspunkte für eine höhere Konfidenz im Bewertungsergebnis.

00.

Agenda

Die Softwarearchitektur eures Systems in Eigenregie bewerten

01. **Warum leichtgewichtig bewerten?**

02. **Verstehe, was Dich speziell macht**

03. **Durchleuchte die Architektur**

04. **Erhöhe die Konfidenz (bei Bedarf)**

05. **Weitere Informationen**

01.

Warum leichtgewichtig bewerten?

Was ist Softwarearchitektur?

Softwarearchitektur :=

Σ wichtige Entscheidungen

wichtig =
- fundamental (betrifft viele)
- im weiteren Verlauf nur schwer zu ändern
- entscheidend für den Erfolg Eures Softwaresystems

Themen für Entscheidungen

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, ...

Was ist ein Review?

Wörtlich:
Review == etwas **(über)prüfen**, besprechen, ...

Gegenstand („etwas“) in unserem Fall:
Die Architektur(-entscheidungen) eines Softwaresystems

Typischer Begriff in der Fachwelt / -literatur auch:
Architekturbewertung (englisch „Evaluation“)

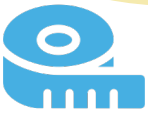
Kann durch Außenstehende erfolgen – muss aber nicht.

Kernelemente eines Reviews

Für eine Bewertung oder ein Review braucht es mindestens

Einen Anlass
Warum bewerten wir?

Einen Gegenstand
Was bewerten wir?



Einen Maßstab
Wonach (wo gegen) bewerten wir?

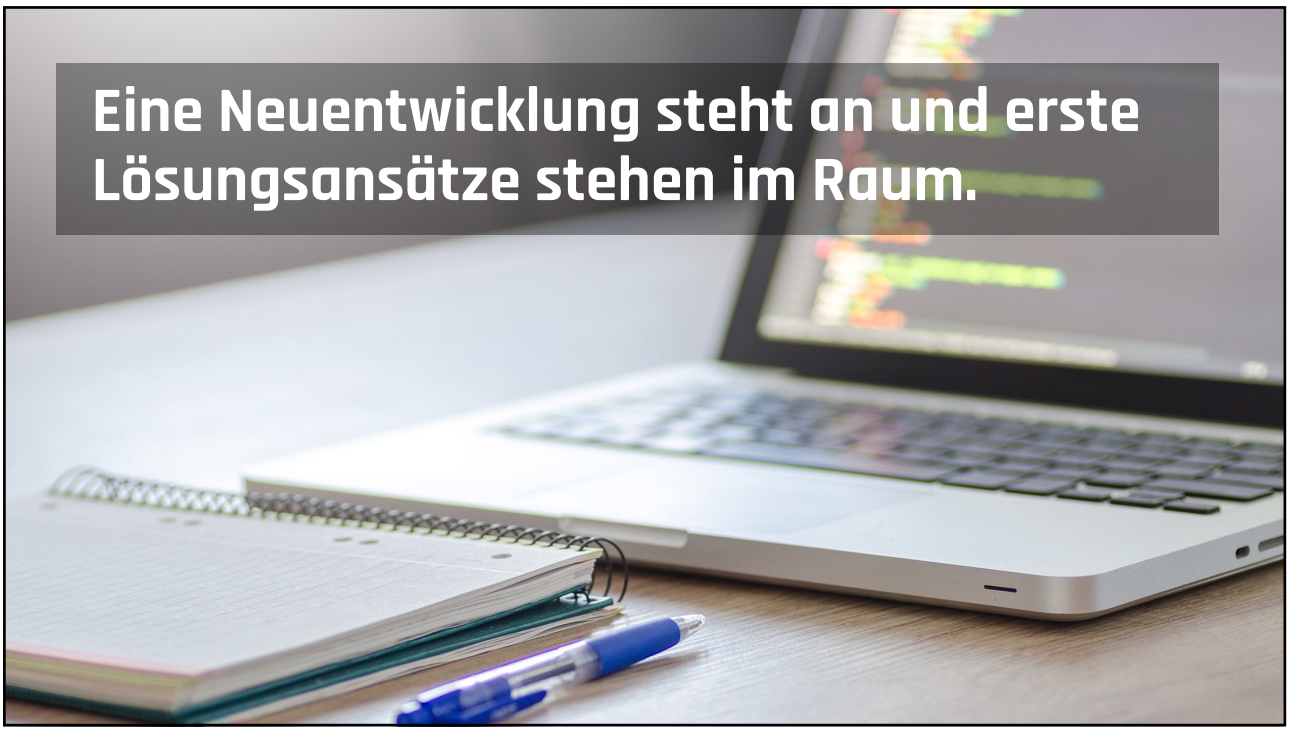
Kernelemente eines Reviews

Für eine Bewertung oder ein Review braucht es mindestens

Einen Anlass
Warum bewerten wir?

Einen Gegenstand
Was bewerten wir?

Einen Maßstab
Wonach (wo gegen) bewerten wir?



Typische Anlässe (... findet Ihr Euch wieder?)

Anlass 1 („Neuanfang“)

Eine **Neuentwicklung** steht an und **erste Lösungsansätze** stehen im Raum.

Leitfrage:
Seid ihr und euer Team auf dem **richtigen Weg**?

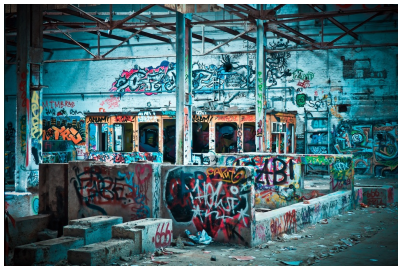
Typische Anlässe (... findet Ihr Euch wieder?)

Anlass 2 („Wünsche“)

Unterschiedliche Stakeholder verfolgen **widersprüchliche Ziele** mit eurer Software.

Leitfrage:
Wie **konkretisiert** und **priorisiert** ihr deren Wünsche?

Typische Anlässe (... findet Ihr Euch wieder?)



Anlass 3 („Unruhe“)

Das **Management** hat das **Vertrauen** in eure Lösung verloren.

Leitfrage:
Wie gewinnt ihr es zurück und strahlt **Sicherheit** aus?

Typische Anlässe (... findet Ihr Euch wieder?)

Anlass 4 („Legacy“)

Größere **Umbaumaßnahmen** in eurer Software stehen an.

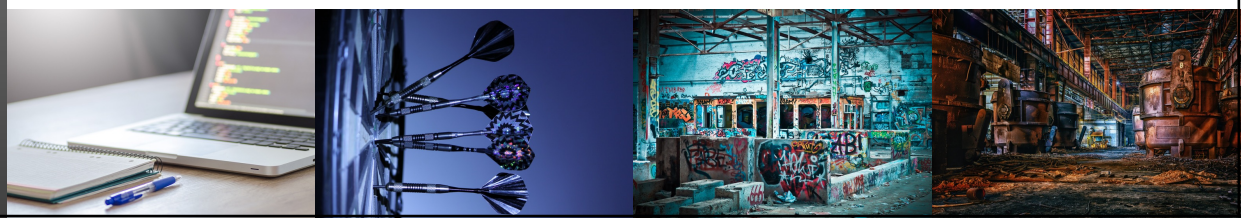
Leitfrage:
Wie wählt ihr passende Lösungsansätze **nachvollziehbar** aus?

Das Leistungsversprechen von Reviews

In all diesen Situationen unterstützen Review-Ansätze und helfen euch und eurem Team die Leitfragen zu beantworten.

Konkret: Software-Reviews …

- … **decken** Kompromisse und **Risiken** von Softwarelösungen **auf**.
- … **sichern** technische und architektonische **Ideen ab**.



Grundsätzliche Ansätze

Quantitative Analyse

Setzt auf Messungen
und Metriken.
In der Regel **Tool-basiert**.

Qualitative Analyse

Setzt auf Diskussion,
Austausch und Durchsprachen.
Oft **Workshop-basiert**.

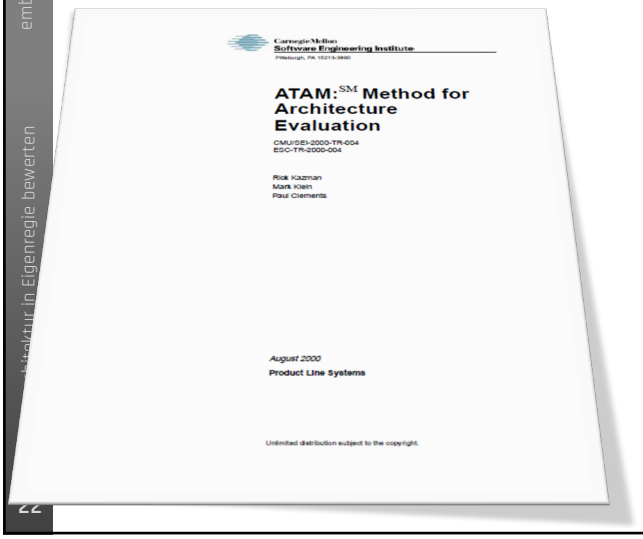
Bewertungsmethoden (Auswahl)

SAAM Software Architecture Analysis Method	**ARID** Architecture Review for Intermediate Designs
ATAM Architecture Tradeoff Analysis Method	**DCAR** Decision Centric Architecture Review
CBAM Cost-Benefit Analysis Method	**DASE** Decision and Scenario based architecture evaluation
TARA Tiny Architecture Review Approach	**Pre-Mortem** Risk-Brainstorming and Mitigation
PBAR Pattern Based Architecture Review	**LASR** Lightweight Approach for Software Reviews

Bewertungsmethoden (Auswahl)

SAAM Software Architecture Analysis Method	**ARID** Architecture Review for Intermediate Designs
ATAM Architecture Tradeoff Analysis Method	**DCAR** Decision Centric Architecture Review
CBAM Cost-Benefit Analysis Method	**DASE** Decision and Scenario based architecture evaluation
TARA Tiny Architecture Review Approach	**Pre-Mortem** Risk-Brainstorming and Mitigation
PBAR Pattern Based Architecture Review	**LASR** Lightweight Approach for Software Reviews

ATAM



Architecture Tradeoff Analysis Method

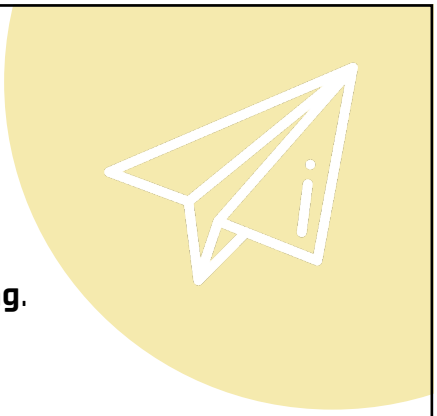
- **Akademischer Ursprung**: Carnegie Mellon University, 2000
- Bekannteste Methode zur Bewertung von Softwarearchitektur
- **Qualitativer** Ansatz, Szenarien- und **Workshop**-basiert
- Früh anwendbar, geht von den **Zielen** aus

Herausforderungen …

Die Anwendung fundierter Bewertungsmethoden ist mitunter schwierig.

- Der Einsatz erfordert häufig **viele Beteiligte**.
- Es sind oftmals **Vorarbeiten nötig**, beispielsweise die Aufbereitung der Geschäftsziele und das Anfertigen eines Architekturüberblicks.
- Sie liefern **nur Rohergebnisse**, die für eine effiziente Kommunikation aufwendig nachzubearbeiten sind.
- Durchführung und Moderation verlangen einiges ab – die Methoden **unterstützen** dabei **wenig**.

Was ist leichtgewichtig?



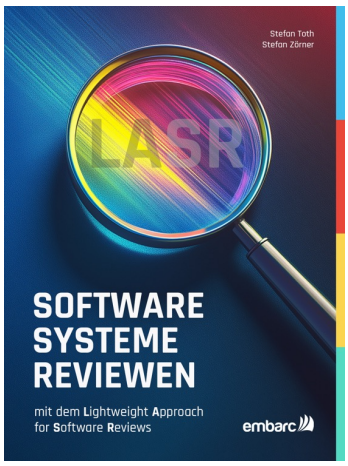
Merkmale eines leichtgewichtigen Reviews

- Die **Anzahl** der Beteiligten / **Stakeholder** ist **gering**.
- **Aufwand** und Dauer sind **überschaubar**.
- **Ergebnisse** liegen vergleichsweise **schnell** vor.

Konkret z.B.

- Entwicklungsteam führt Review **allein** und **ohne Vorbereitung** durch.
- Bereits nach einem halben Tag liegt ein **kommunizierbares Ergebnis** vor.

Erfahrungswissen



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

Leanpub

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

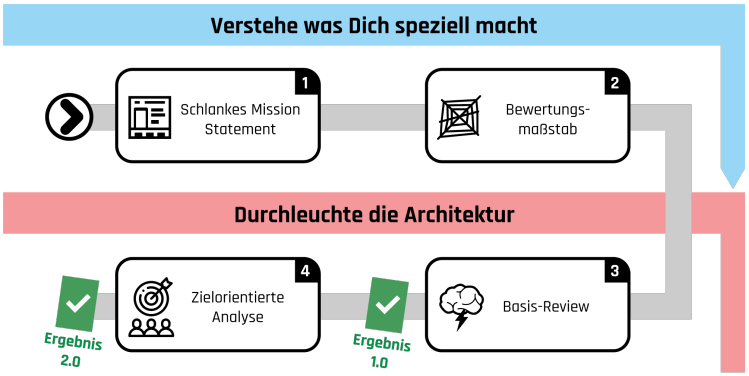
➔ leanpub.com/software-systeme-reviewen/

Ein schlanker Ansatz

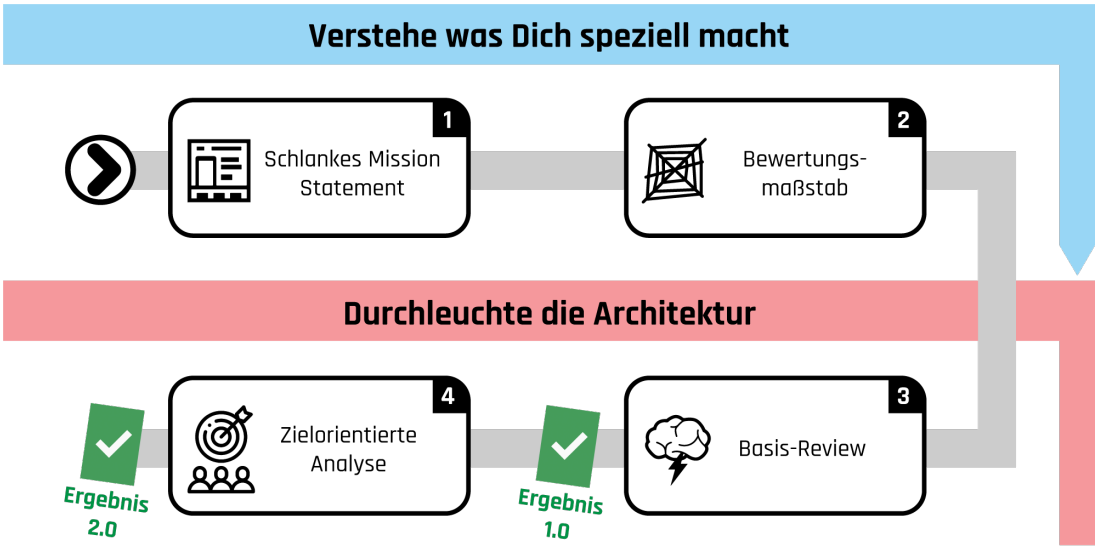
LASR (Lightweight Approach for Software-Reviews) ist ein strukturiertes Vorgehen für leichtgewichtige Software-Reviews.

Ablauf LASR-Review

- 2 Tätigkeiten
- 4 Schritte
- Ein frühes erstes Ergebnis



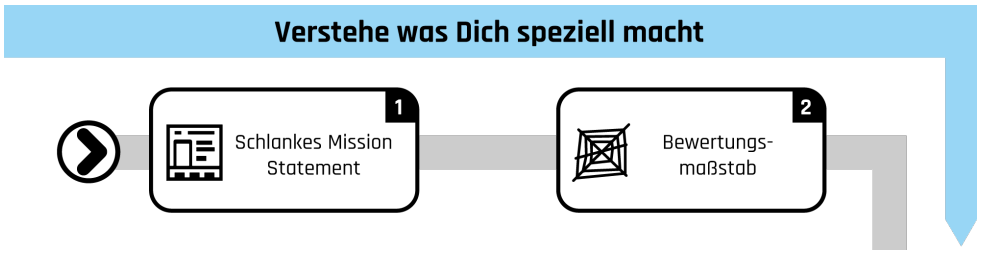
Tätigkeiten und Schritte im LASR-Review



02.

Verstehe, was Dich speziell macht

Verstehe, was Dich speziell macht



Was ist zu tun?

- Die Aufgabenstellung für das System zu einem prägnanten Mission Statement verdichten
- Die Top 3-5 Ziele des Systems identifizieren
- Jeweiliges Ziel-Level festlegen

Kernelemente eines Reviews

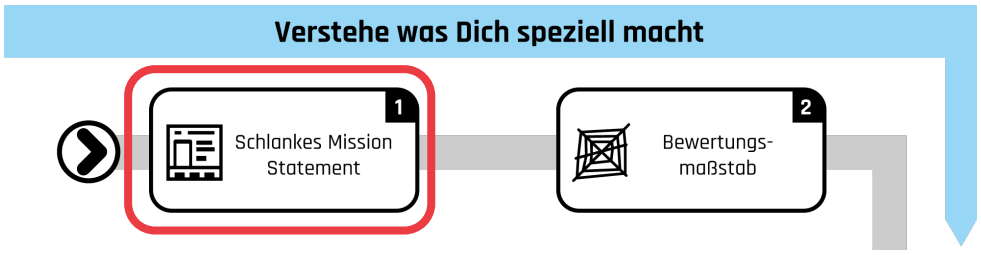
Für eine Bewertung oder ein Review braucht es mindestens

Einen Anlass
Warum bewerten wir?

Einen Gegenstand
Was bewerten wir?

Einen Maßstab
Wonach (wo gegen) bewerten wir?

Verstehe, was Dich speziell macht



Was ist zu tun?

- **Die Aufgabenstellung für das System zu einem prägnanten Mission Statement verdichten**
- Die Top 3-5 Ziele des Systems identifizieren
- Jeweiliges Ziel-Level festlegen

(1) Schlankes Mission Statement

Fokussiert und **grenzt** das zu betrachtende System **ab** durch Finden sogenannter "Claims" (Ansprüche) der Software.

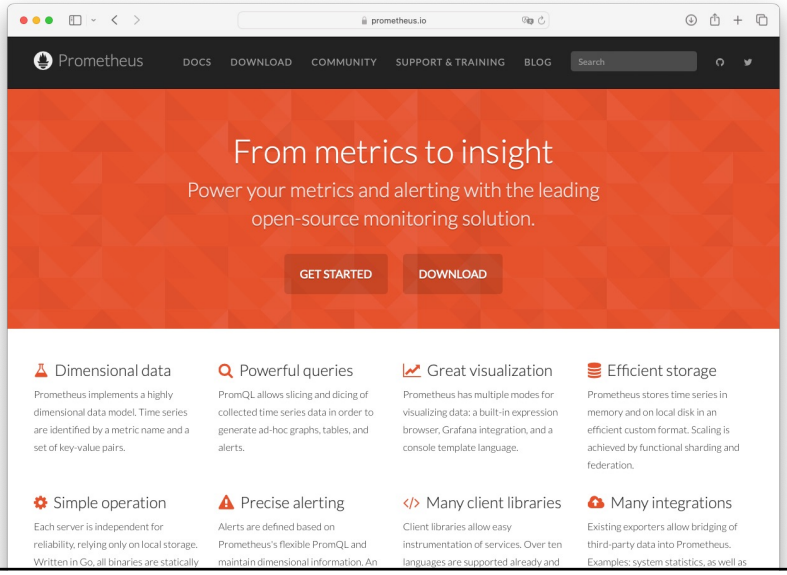
Methodischer Ansatz

Sammeln der Claim(s) mit **Landing Page-Metapher**

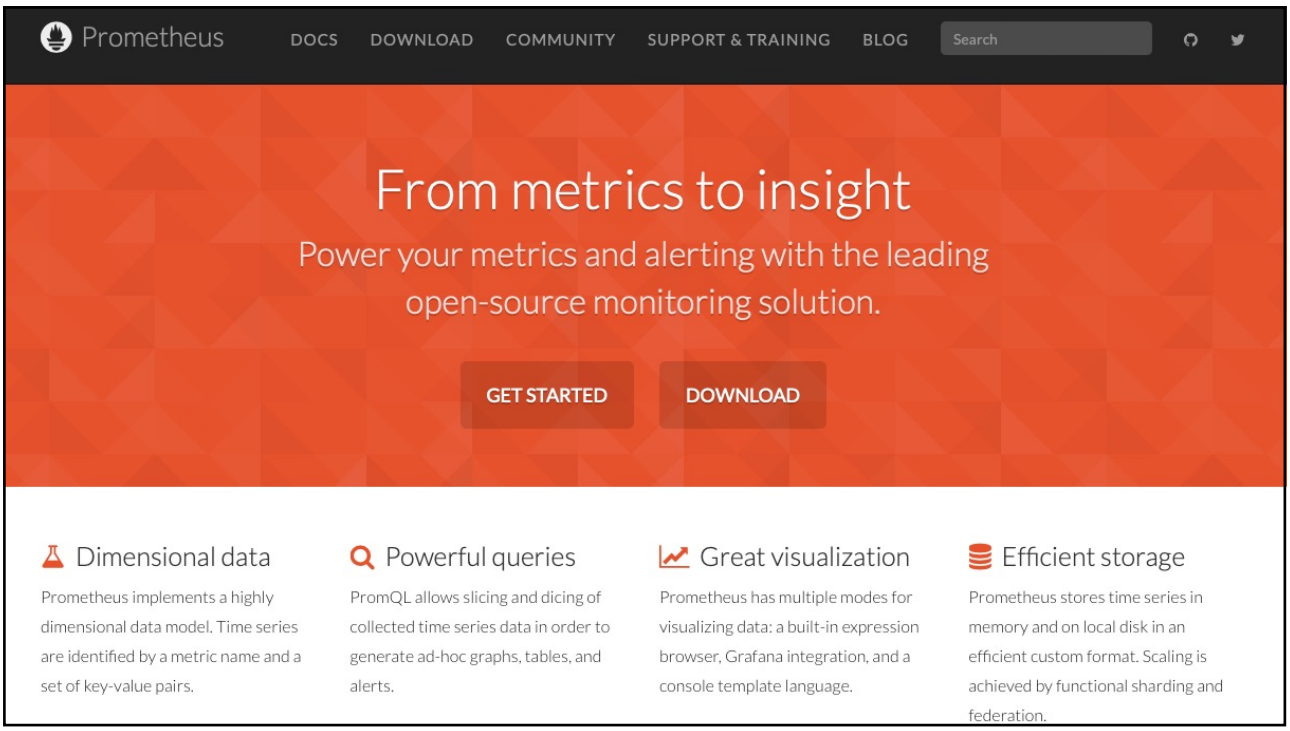
Ergebnis des Schrittes

Prägnante Produktidee (Stichpunkte)

Beispiel: Landing Page von Prometheus

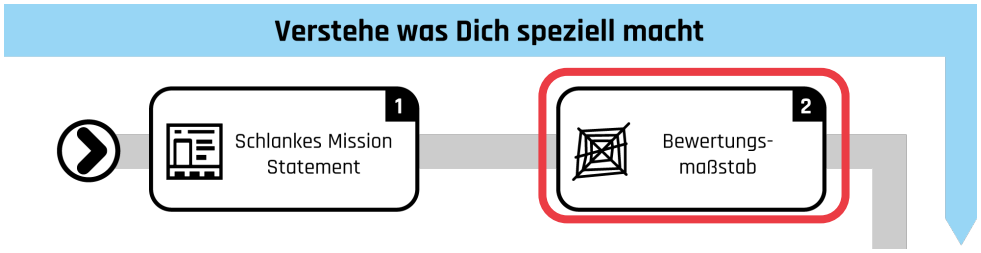


→ prometheus.io


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.


embarc.de
Softwarearchitektur in Eigenregie bewerten
35
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.
Keine Datensammelei
keine Angabe persönlicher Informationen erforderlich
Ende-zu-Ende-verschlüsselung
Open-Source
Threema
Mobile-App verfügbar für Android und iOS-Geräte
Schutz der Privatsphäre stets gewährleistet
zuverlässiger Versand der Nachrichten und Medien
datenschutzkonform (DSGVO, etc.)
Identität eines Kommunikationspartners zweifelsfrei feststellbar
sicher mit einzelnen Kontakten und in Gruppen kommunizieren
Beispiel im digitalen Whiteboard (Mural)
Quelle: S. Toth, S. Zörner: „Software-Systeme reviewen“, Leanpub 2023

Verstehe, was Dich speziell macht



Was ist zu tun?

- Die Aufgabenstellung für das System zu einem prägnanten Mission Statement verdichten
- **Die Top 3-5 Ziele des Systems identifizieren**
- Jeweiliges Ziel-Level festlegen

(2) Bewertungsmaßstab

Identifiziert und priorisiert grob die entscheidenden **Qualitätsmerkmale**.

Methodischer Ansatz

Top-5-Challenger zur Zielauswahl

Ergebnisse des Schrittes

3-5 Qualitätsziele inklusive grober Ziel-Einschätzung

Qualitätsmerkmale (Begriffe nach ISO 25010:2023)

Funktionale Eignung
(Functional Suitability)
Sind die berechneten Ergebnisse genau genug / exakt, ist die Funktionalität angemessen? ...

Effizienz
(Performance Efficiency)
Antwortet die Software schnell, hat sie einen hohen Durchsatz, einen geringen Ressourcenverbrauch? ...

Kompatibilität
(Compatibility)
Ist die Software konform zu Standards, arbeitet sie gut mit anderen zusammen? ...

Benutzbarkeit
(Interaction Capability)
Ist die Software intuitiv zu bedienen, wiedererkennbar, leicht zu erlernen, attraktiv? ...

Zuverlässigkeit
(Reliability)
Ist das System verfügbar, tolerant gegenüber Fehlern, nach Abstürzen schnell wieder hergestellt? ...

Sicherheit
(Security)
Ist das System sicher vor Angriffen? Sind Daten und Funktion vor unberechtigtem Zugriff geschützt? ...

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

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

Betriebssicherheit
(Safety)
Sind Personen, Tiere, Sachen und Umwelt vor Schäden durch die Software geschützt? ...

Kernelemente eines Reviews

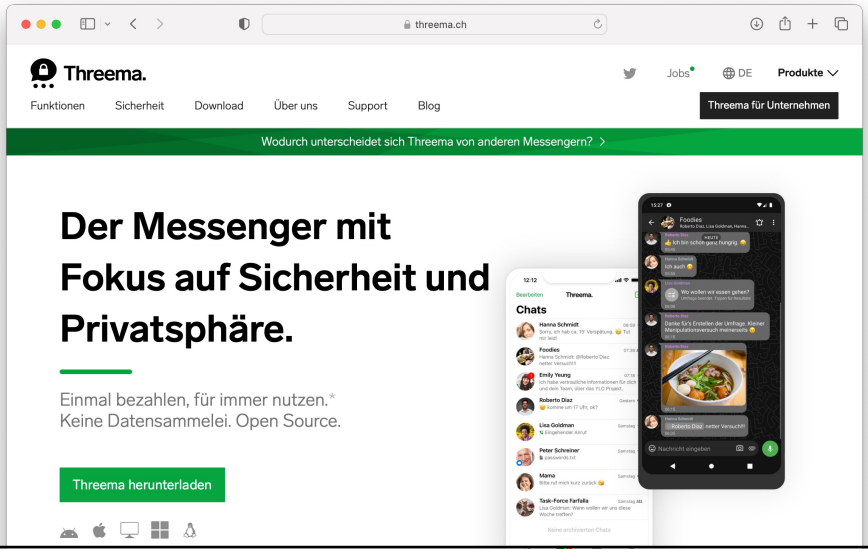
Für eine Bewertung oder ein Review braucht es mindestens

Einen Anlass
Warum bewerten wir?

Einen Gegenstand
Was bewerten wir?

Einen Maßstab
Wonach (wo gegen) bewerten wir?

Weiteres Beispiel für eine Startseite



Die Qualitätsziele von Threema

Sicherheit	Kommunikations- und Nutzerdaten sind geschützt.
Benutzbarkeit	Einfach zu verwenden.
Zuverlässigkeit	Zuverlässig und effizient im Betrieb.
Kompatibilität	Interoperabel über alle Plattformen hinweg.
Wartbarkeit	Leicht erweiterbar und anpassbar.

Quelle: „Architektur-Porträt: Threema“, Stefan Zörner 2023

Top-5-Challenger (Unterstützungsmaterial)

Material:

- 14 Karten mit Qualitätsmerkmalen

Vorbereitung:

- Mischt die Karten in einem verdeckten Stapel.
- Deckt 5 Karten zufällig auf und legt sie nebeneinander. (das ist Eure Start Top-5)
- Legt die übrigen 9 in einer 3 x 3 Matrix für alle gut sichtbar offen aus.

Top-5-Challenger, Vorbereitung (Beispiel)

Start Top-5

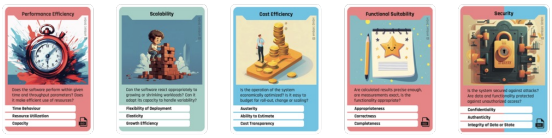
Kandidaten

Ablauf einer Runde

Einen „Challenger“ nominieren

- Team-Mitglied schlägt Ziel vor, das in Top-5 fehlt.
- Die Gruppe diskutiert darüber.
- Der Challenger verdrängt eine Karte oder landet selbst auf dem Ablagestapel.

Nach 2 Runden (Beispiel)

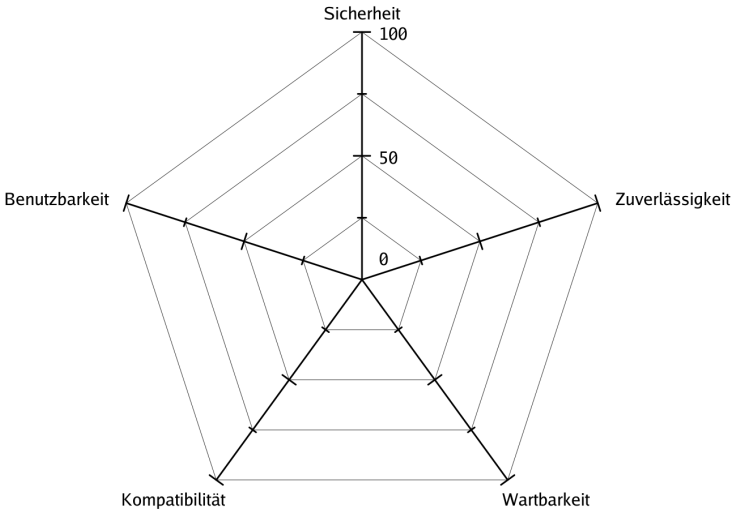


Aktuelle Top-5

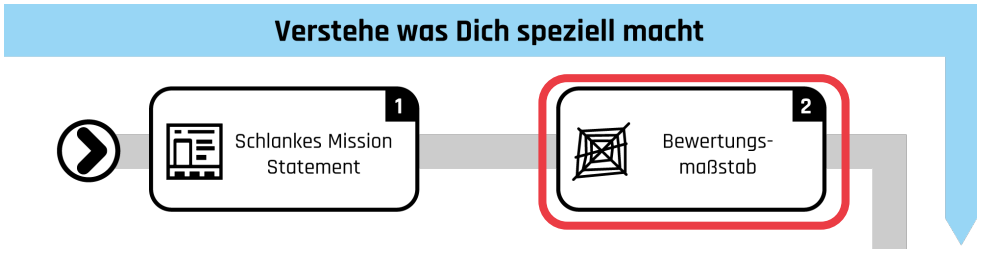
Ablagestapel

Kandidaten

Ergebnis nach diesem Schritt (Beispiel)



Verstehe, was Dich speziell macht

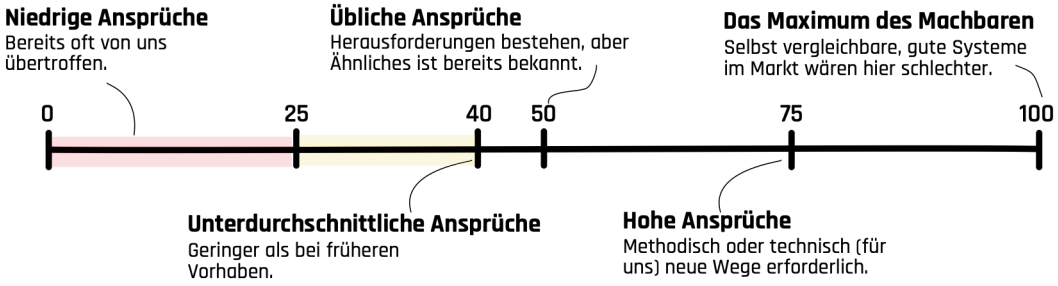


Was ist zu tun?

- Die Aufgabenstellung für das System zu einem prägnanten Mission Statement verdichten
- Die Top 3-5 Ziele des Systems identifizieren
- **Jeweiliges Ziel-Level festlegen**

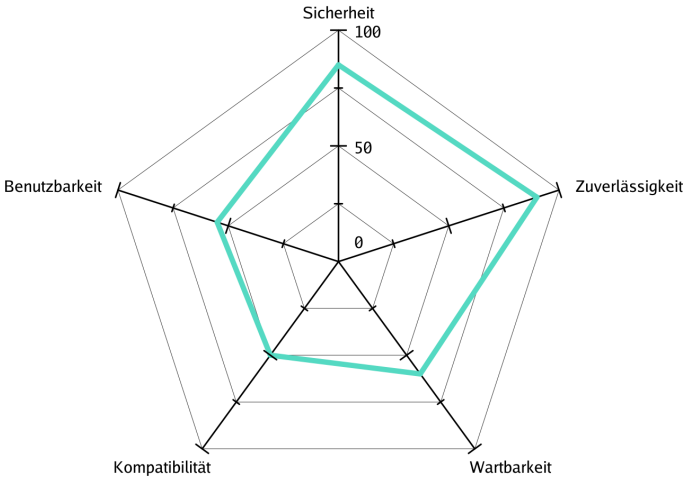
Zielwerte bestimmen

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 einzeichnen



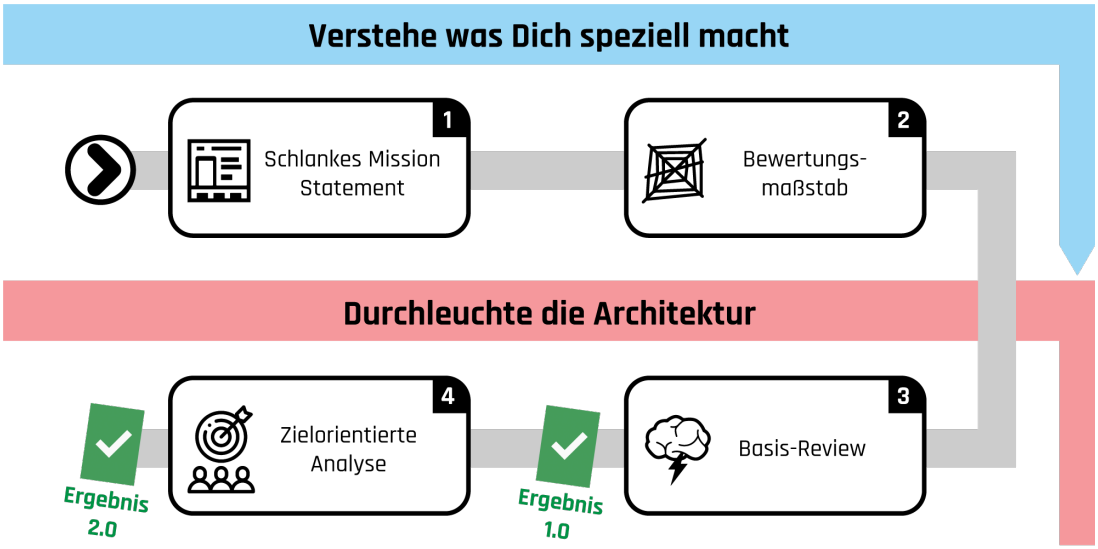
LASR-Ergebnisdiagramm

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

Die Zielwerte spannen darauf eine grüne Linie auf.

Beispiel

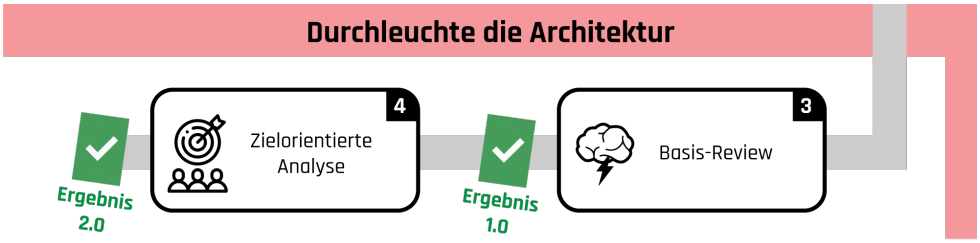
Mit Schritt 2 steht der Ergebnisrahmen.



03.

Durchleuchte die Architektur

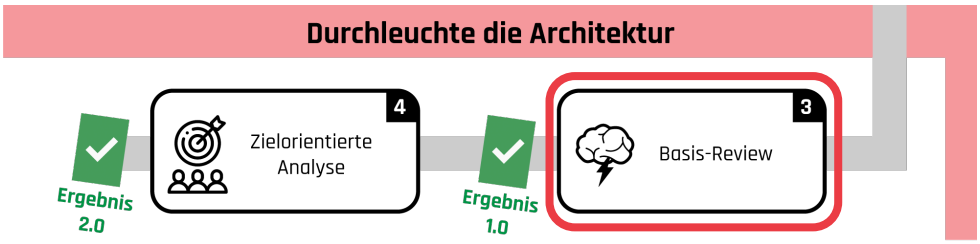
Durchleuchte die Architektur



Was ist zu tun?

- Die größten Risiken des Systems identifizieren
- Die aus den Risiken resultierende Abweichung zu den Zielen quantifizieren („Lücken“)
- Zielorientierte Analysen durchführen, um das Review-Ergebnis bei Bedarf zu schärfen

Durchleuchte die Architektur



Was ist zu tun?

- **Die größten Risiken des Systems identifizieren**
- Die aus den Risiken resultierende Abweichung zu den Zielen quantifizieren („Lücken“)
- Zielorientierte Analysen durchführen, um das Review-Ergebnis bei Bedarf zu schärfen

(3) Basis-Review

Produziert ein erstes Ergebnis in Form konkreter **Risiken**, die der Zielerreichung im Weg stehen.

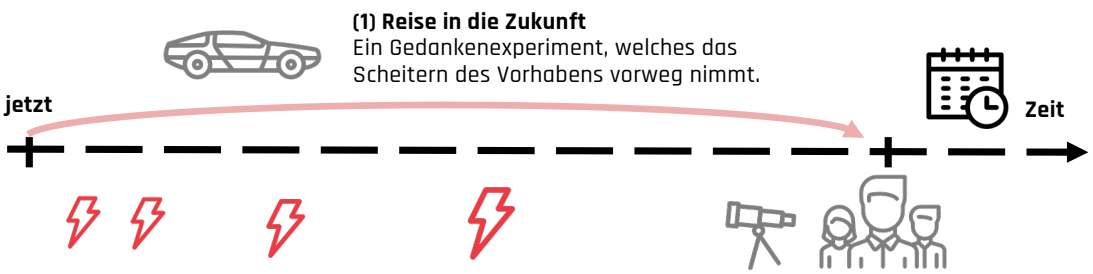
Methodischer Ansatz

Brainstorming, angelehnt an **Pre-Mortem**

Ergebnisse des Schrittes

"Abweichung" vom Ziel, Ist-Einschätzung

Was ist ein Pre-Mortem?



(1) Reise in die Zukunft
Ein Gedankenexperiment, welches das Scheitern des Vorhabens vorweg nimmt.

(2) Blick zurück
Das Team blickt gemeinsam zurück und überlegt, warum das Vorhaben gescheitert ist.

(3) Gründe des Scheiterns/Risiken
Probleme unterschiedlichster Art und Größe machen das Scheitern „plausibel".

Die 8 Risiko-Kategorien von LASR

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

2 Kompetenz und Erfahrung
- Fehlendes oder isoliertes Wissen
- Zu kleine Lernfenster
- Fehlendes Prozessverständnis
- Tool-Probleme
- Schwere Schätzbarkeit
- ...

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

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

5 Altsysteme und Altlasten
- Legacy-Behinderungen
- Innovationsstau
- Zerbrechliche Lösungen
- Fehlende Tests
- Wenig Dokumentation
- Fehlendes Verständnis
- ...

6 Organisation und Prozesse
- Geschwollene Prozesse
- Behindernde Rollenmodelle
- Organisationsgrenzen
- Politik
- Einengende Standards
- Isolierte Entscheidungskompetenzen
- ...

7 Betrieb und Deployment
- Blockierende Prozesse
- Geringe Automatisierung
- Wenig Feedback
- Fehlende Betriebs- / Ausfallkonzepte
- Tool-Probleme
- ...

8 Weiche Faktoren
- Uneinigkeiten
- Konflikte
- Fehlende Disziplin
- Kommunikationsbarrieren
- Rollenthematiken
- Unpassende Kultur
- ...

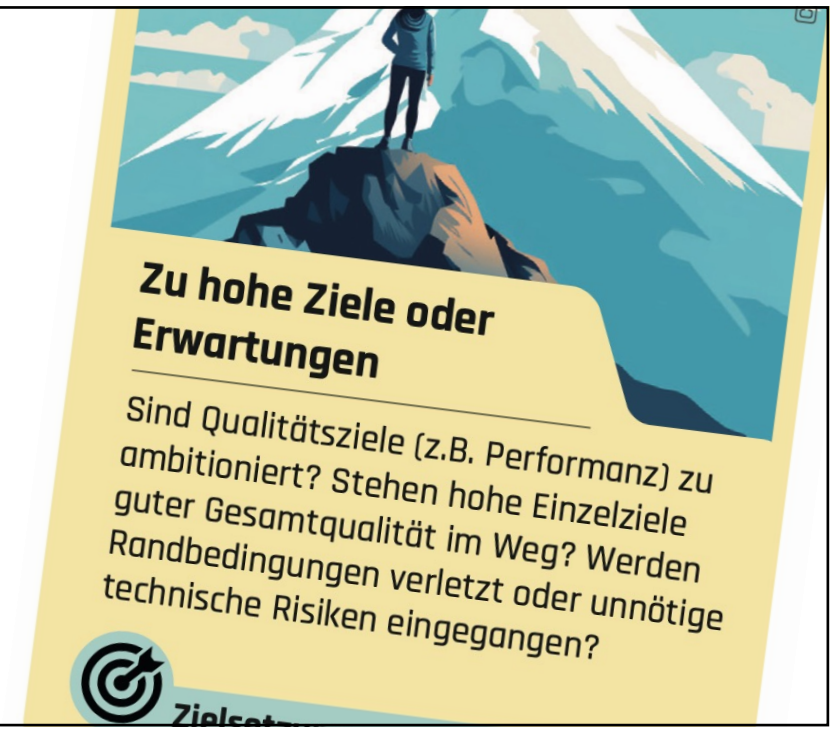
Typische Risikothemen als Ideengeber

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

Disclaimer

- Die folgenden **Beispiel-Risiken** sind **realistisch** für Dinge, die Team-Mitglieder im Rahmen eines Risiko-Workshops im Pre-Mortem aufwerfen.
- Bezogen auf unser Beispiel **Threema** hier sind sie **rein fiktiv**. Sie dienen der Illustration.

Beispielkarte



Hypothetisches Risiko (1)

Aufhebung des Gruppen-Limits

„Wir haben die Begrenzung in Gruppen-Größen aufgehoben. Unsere User sind begeistert von den neuen Möglichkeiten. Später bricht der Chat-Server im Threema-Backend unter der massiven Last und der Menge zu speichernden Nachrichten zusammen."

Hypothetisches Risiko (2)

Das Electron-Framework fällt weg

„Wir waren gezwungen das Electron-Framework in unserer Desktop-App durch eine Alternative / etwas Selbstgebautes zu ersetzen. Das zieht sich. Zwischenzeitlich haben wir den Download des Desktop-Clients von der Seite genommen, weil er auf aktuellen OS nicht mehr funktioniert. “

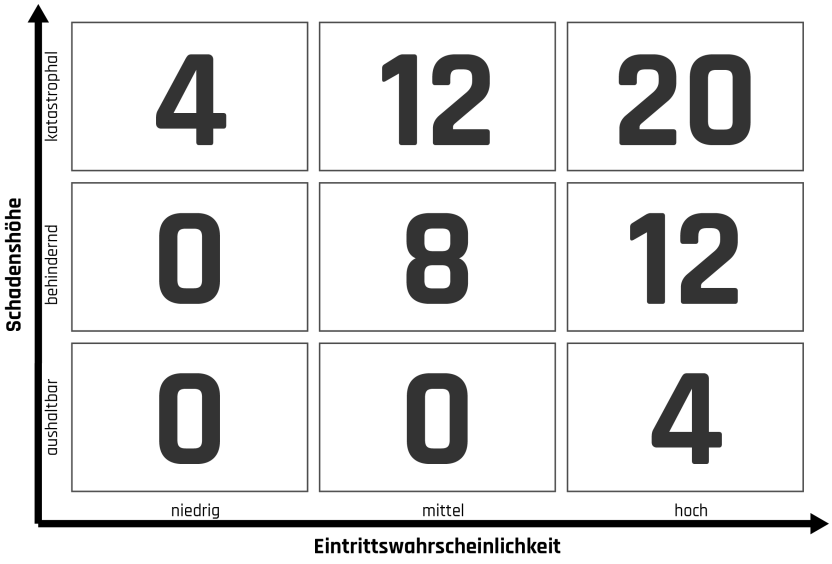
Hypothetisches Risiko (3)

Private Kommunikation scannen

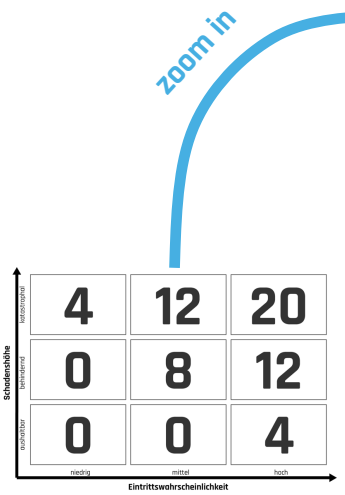
„Die europäische Gesetzgebung hat Messenger-Anbieter verpflichtet, sämtliche private Kommunikation und Dateien zu durchleuchten. Threema muss daraufhin die Lösung für private Nutzer komplett neu denken. “

Risiken einordnen

?



Den betroffenen Zielen zuordnen



Risikobereich mittel/hoch

mittlere Wahrscheinlichkeit dass das Risiko zum Problem wird

hoher Schaden für zumindest ein Qualitätsziel (katastrophal)

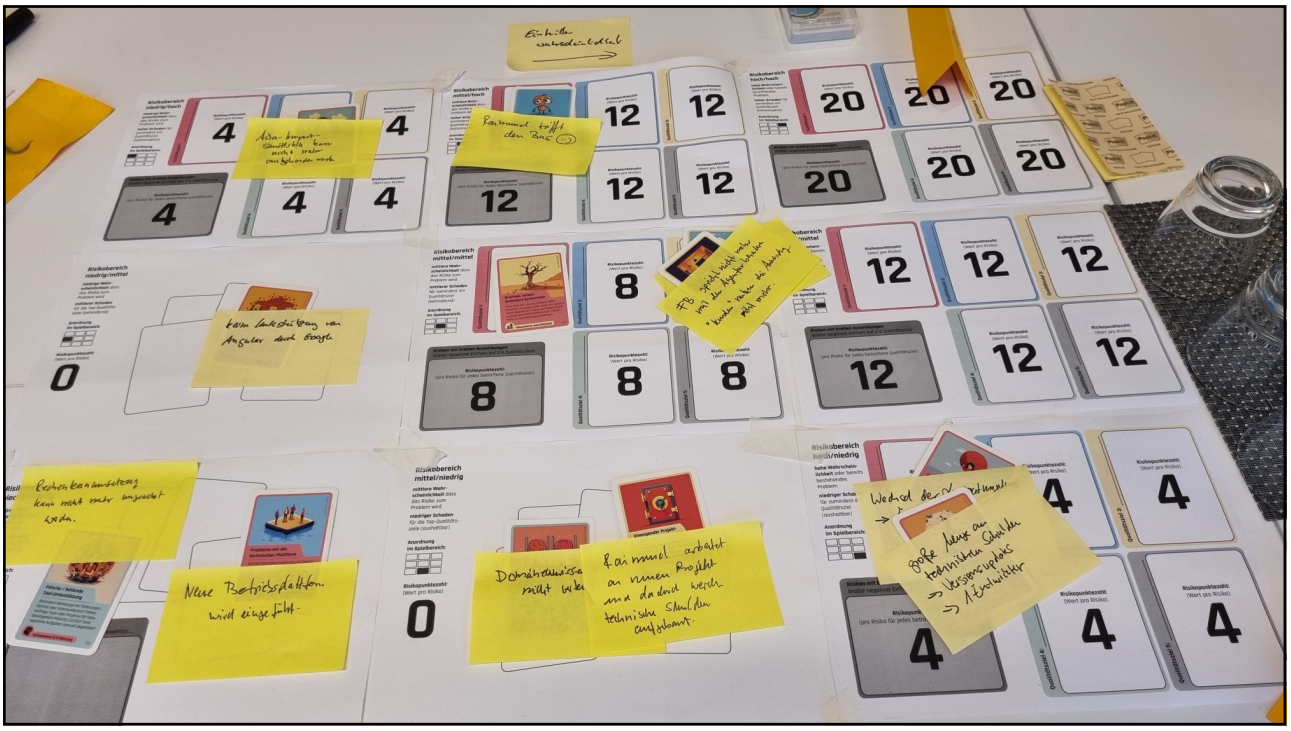
Anordnung im Spielbereich:

Risiken mit breiten Auswirkungen
Großer negativer Einfluss auf 2-5 Qualitätsziele

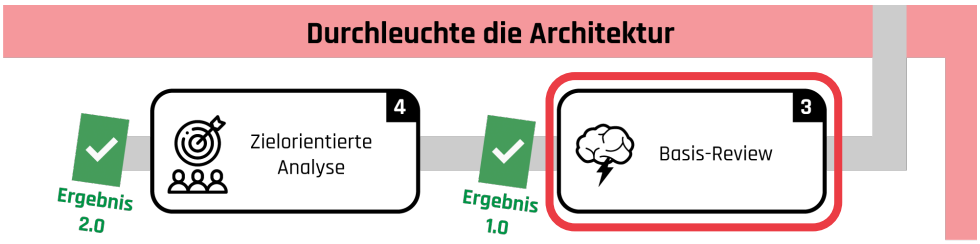
Risikopunktezahl:
(pro Risiko für jedes betroffene Qualitätsziel)
12

Qualitätsziel 1:	Qualitätsziel 2:	Qualitätsziel 3:
Risikopunktezahl: (Wert pro Risiko) **12**	Risikopunktezahl: (Wert pro Risiko) **12**	Risikopunktezahl: (Wert pro Risiko) **12**

Qualitätsziel 4:	Qualitätsziel 5:
Risikopunktezahl: (Wert pro Risiko) **12**	Risikopunktezahl: (Wert pro Risiko) **12**



Durchleuchte die Architektur



Was ist zu tun?

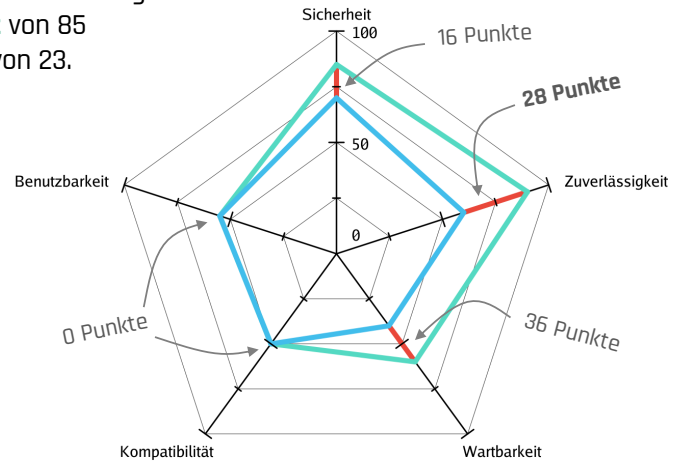
- Die größten Risiken des Systems identifizieren
- **Die aus den Risiken resultierende Abweichung zu den Zielen quantifizieren („Lücken“)**
- Zielorientierte Analysen durchführen, um das Review-Ergebnis bei Bedarf zu schärfen

Von Risiken zu „Lücken“

28% Abweichung
vom **Ziellevel**
auf der **Qualitätszielachse**

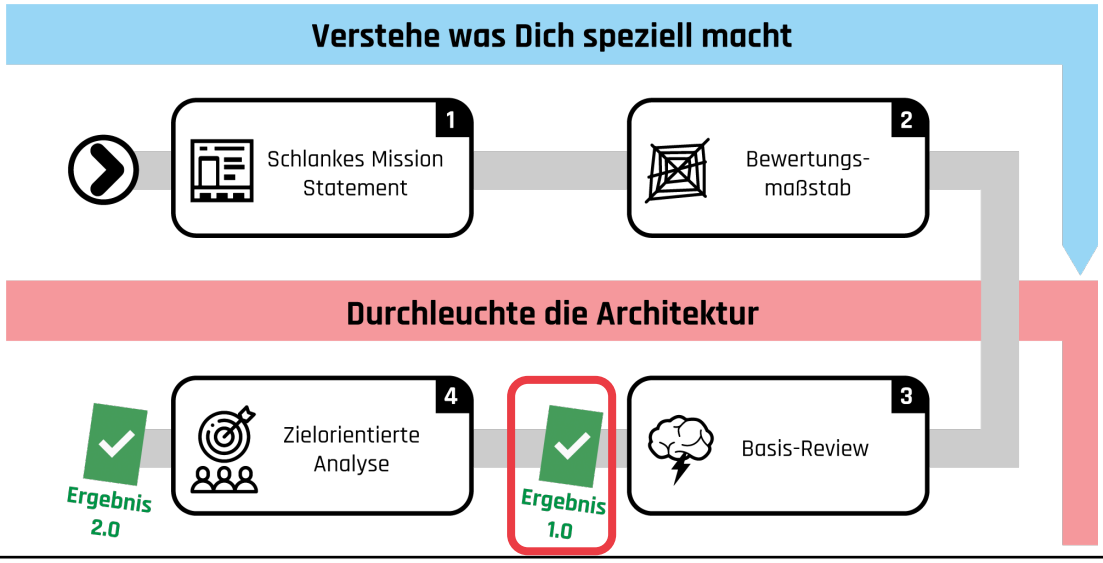
Abweichungen einzeichnen

Beispiel
28 Prozent-Punkte Abweichung bei einem Zielwert von 85 ergibt eine Lücke von 23.



- Ziellinie
- Abweichungen (Lücken)
- Ergebnislinie

Basis-Review, 1. Ergebnis nach Schritt 3



Ergebnis 1.0

LASR Ergebnis 1.0

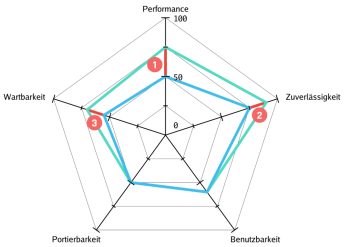
Symbolhaft dargestellte Elemente des Ergebnisses von LASR nach Schritt 3 (Basis-Review)

Mission Statement

[Ein kurzer Text (oder Slogan) zur gebauten Lösung, der die wichtigsten Merkmale und Eigenschaften der Software vermittelt und zentrale qualitative Anforderungen anklingen lässt - ggf. als Set von „Claims" aus einem Brainstorming.]

Claim Claim Claim Claim

LASR Ergebnisdiagramm (Spinnennetzgrafik)



Lücken und Risiken

1 [Kurze Lücken-Beschreibung + ausgewählte Details]

Risiko (2C) Risiko (3B) Risiko (2B) Risiko (1C)

2 [Kurze Lücken-Beschreibung + ausgewählte Details]

3 [Kurze Lücken-Beschreibung + ausgewählte Details]

Stärken und Schwächen

Ansatz (+)	[kurze Beschreibung]

Ansatz (-)	[kurze Beschreibung]

LASR an einem Vormittag

Review-Workshop System YX -- Agenda

Die Zeiten dienen Euch zur groben Orientierung.
Wir machen aber pünktlich Schluss.

09:00 - 10:00	10:15 - 11:30	11:45 - 12:15
Begrüßung, Einstieg LASR — ca. 10 Minuten	3 Basis-Review ("Pre-Mortem") — ca. 60 Minuten	3 Ergebnisse diskutieren — ca. 10 Minuten
1 Abgrenzung, Mission Statement ("Claims") — ca. 15-20 Minuten	3 Zielerreichung betimmen — ca. 15 Minuten	3 Ziel-orientierte Analyse planen, TODOs — ca. 10-20 Minuten
2 Bewertungsmaßstab festlegen — ca. 20-90 Minuten	kurze Pause	Ende des Workshops
kurze Pause (+ Basis-Review vorbereiten)		

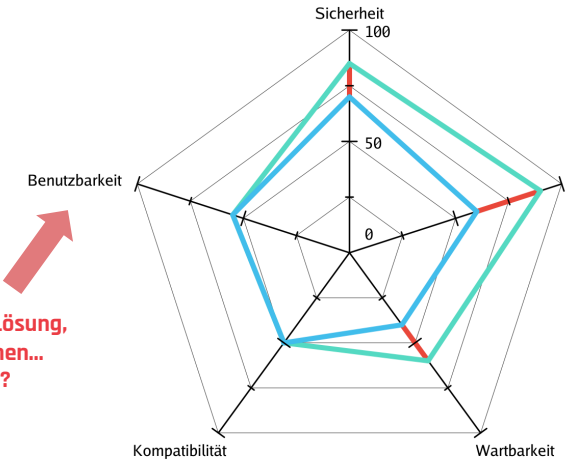
(c) embarc GmbH

Beispiel-Agenda

Quelle: S. Toth, S. Zörner: „Leichtgewichtige Software-Reviews mit LASR ", Informatik Aktuell 2024

04. Erhöhe die Konfidenz (bei Bedarf)

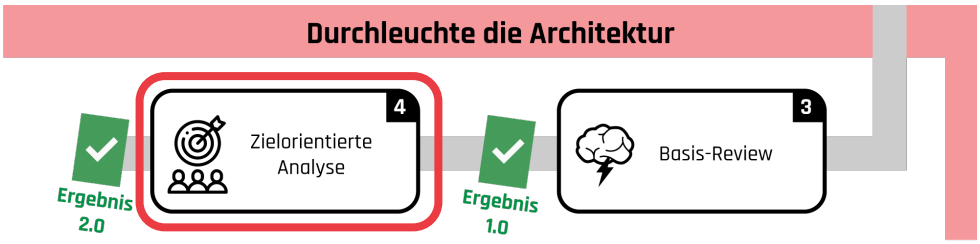
Fokus: Zielorientierte Analyse (Beispiele)



Uneinigkeit!
Lücke ist strittig

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

Durchleuchte die Architektur



Was ist zu tun?

- Die größten Risiken des Systems identifizieren
- Die aus den Risiken resultierende Abweichung zu den Zielen quantifizieren („Lücken“)
- **Zielorientierte Analysen durchführen, um das Review-Ergebnis bei Bedarf zu schärfen**

(4) Zielorientierte Analyse

Untersucht **gezielt** Stärken und Schwächen im System und **schärft** so **das Ergebnis**.

Methodischer Ansatz

Qualitative Durchsprache "strittiger" Ziel-Achsen

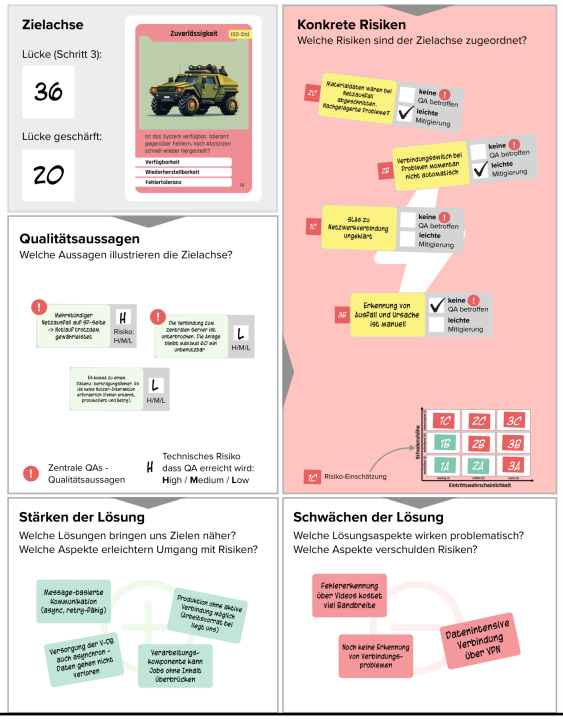
Ergebnisse des Schrittes

verbesserte Ist-Einschätzung, Qualitätsaussagen

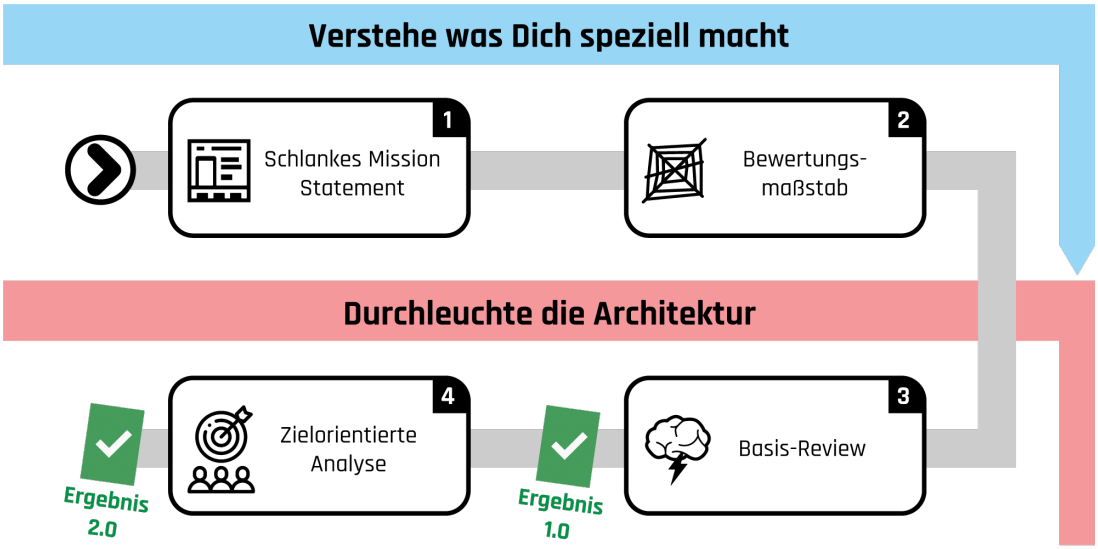
Template

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

Beispiel



Tätigkeiten und Schritte im Kern-Review



Typische Unsicherheiten im Anschluss

Hängt da noch mehr dran?

Gefundene **Risiken** sind in ihrer Natur oder Auswirkung **zu wenig verstanden**, um direkt hilfreiche Aktivitäten daraus abzuleiten.

In der Theorie OK, aber was ist mit …?

Rahmenbedingungen (z.B. Standards) oder die eigene **Organisation** wurde als Risikofaktor **zu wenig betrachtet**.

Wo ist der Code?

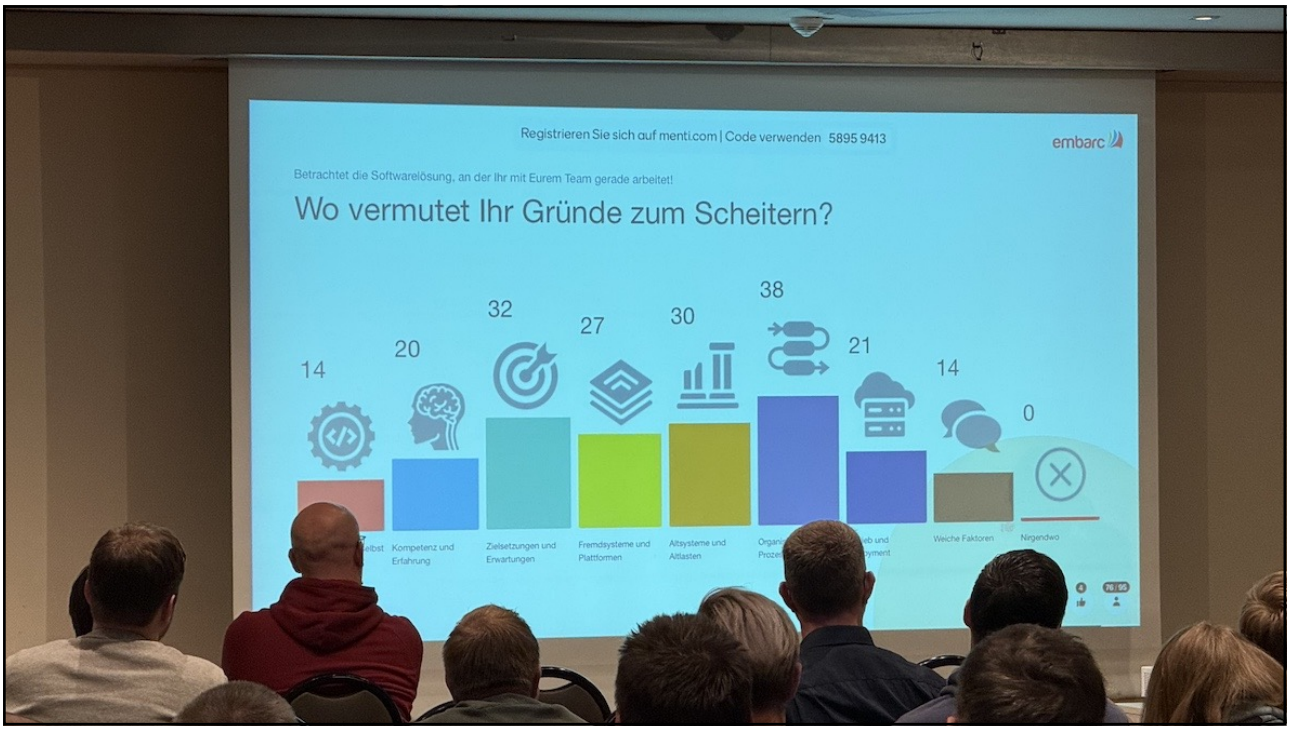
Die **Architekturebene** wird als **zu abstrakt** für die tatsächlichen Probleme des Vorhabens wahrgenommen.

Wo fangen wir an?

Die **Priorisierung** der identifizierten Probleme ist **schwierig**, ggf. fehlt es bei kleinteiligen Ergebnissen an Überblick oder Einordnung.

Ist nicht auch XY wichtig?

Der **Fokus** auf max. 5 Qualitätsziele wirkt **problematisch.**



Wo vermutet Ihr Gründe zum Scheitern?

Wie lange in Produktion	Softwarelösung selbst	Kompetenz und Erfahrung	Zielsetzungen und Erwartungen	Fremdsysteme und Plattformen	Altsysteme und Altlasten	Organisation und Prozesse	Betrieb und Deployment	Weiche Faktoren	Nirgendwo
Noch gar nicht	25%	33%	50%	31%	33%	54%	23%	23%	2%
0-6 Monate	22%	22%	57%	30%	30%	49%	24%	8%	0%
6 Monate bis 2 Jahre	17%	35%	52%	36%	36%	61%	17%	12%	0%
2 bis 5 Jahre	18%	27%	50%	42%	30%	61%	19%	14%	0%
5 - 10 Jahre	13%	32%	42%	38%	53%	67%	18%	13%	0%
10 - 20 Jahre	18%	31%	44%	30%	59%	46%	21%	12%	0%
Länger als 20 Jahre	13%	38%	33%	25%	77%	39%	21%	15%	0%

Skala:					
	0%	15%	25%	50%	> 70%

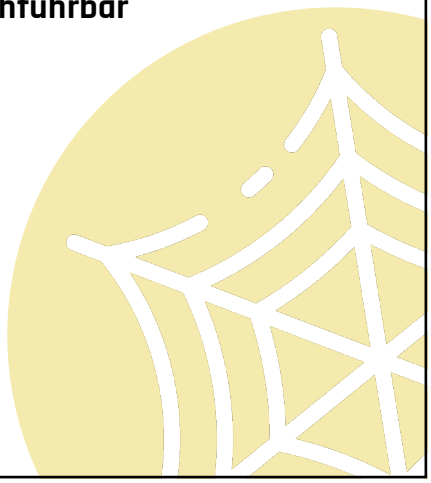
Heatmap aus Kreuztabelle mit Gründen zum Scheitern nach Alter des Softwaresystems.
Quelle: Stefan Zörner: "Gründe zum Scheitern in 500+ Software-Systemen", embarc-Blog, 22.12.2025

05.

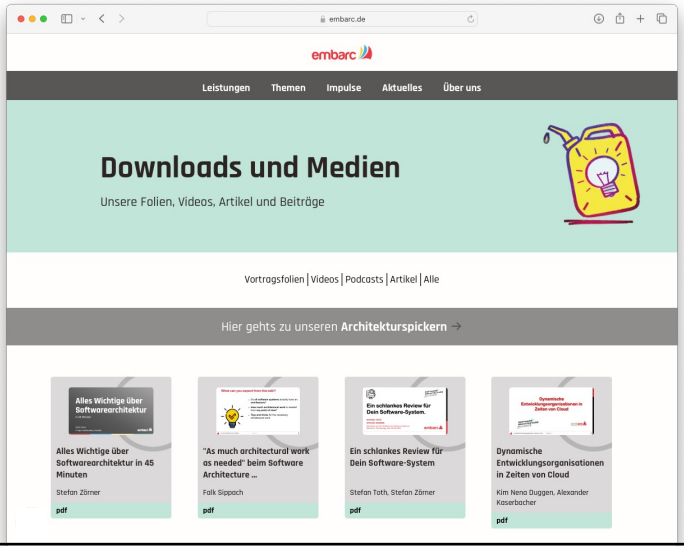
Weitere Informationen

Markante Merkmale von LASR

- Schlankere Methode als ATAM, trotzdem **zielorientiert**
- Mit dem **eigenen Team** und potentiell **alleine durchführbar**
- Liefert **schnell** ein **erstes Ergebnis**, z.B. an einem Nachmittag
- **Spinnennetzgraphik** zur Erkenntnisverdichtung und -kommunikation
- optionale Aktivitäten zur schrittweisen **Konfidenzerhöhung bei Bedarf**
- **Unterstützungsmaterial** für Bewertungsmaßstab, Risikofindung und Ergebniserarbeitung

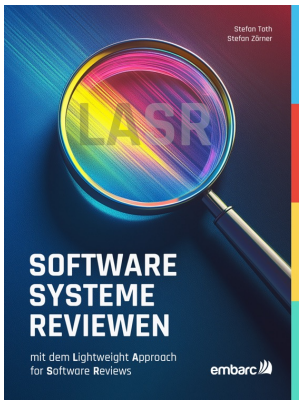


Folien als PDF zum Download



→ embarc.de/download/

Buchtipp zum Thema



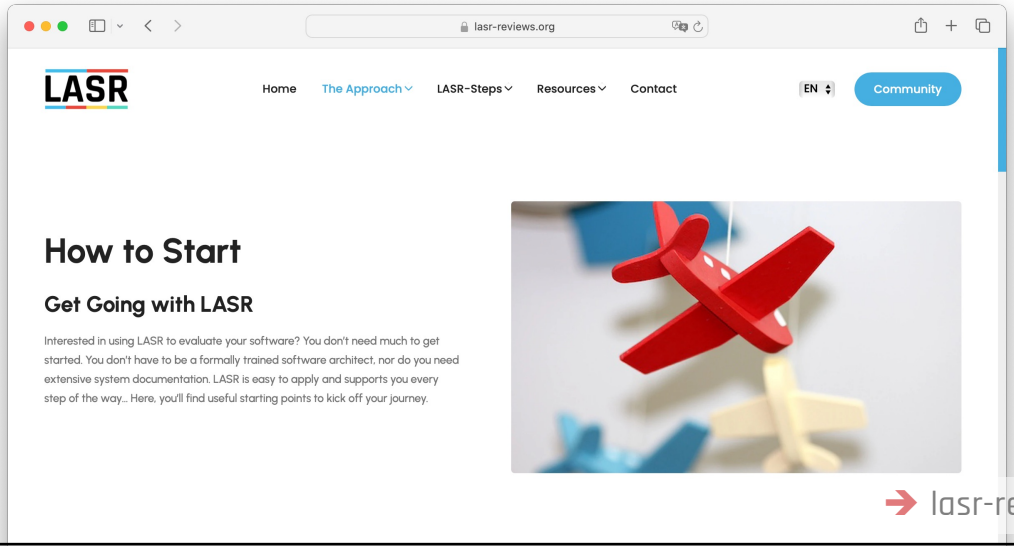
Software-Systeme reviewen
mit dem Lightweight Approach
for Software Reviews - LASR

Leanpub

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

➔ leanpub.com/software-systeme-reviewen/

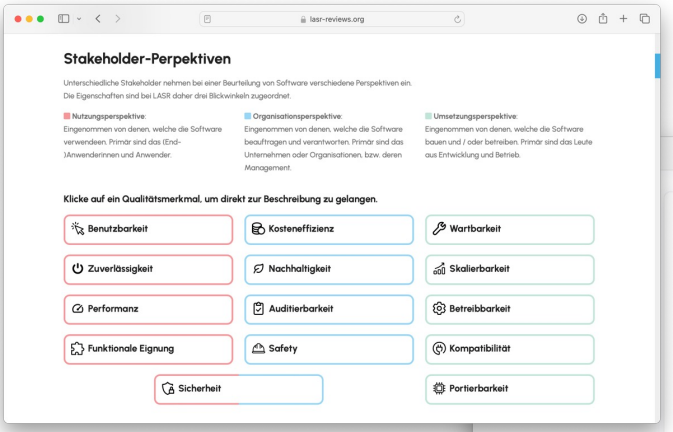
„Offizielle“ LASR-Webseite (EN + DE)

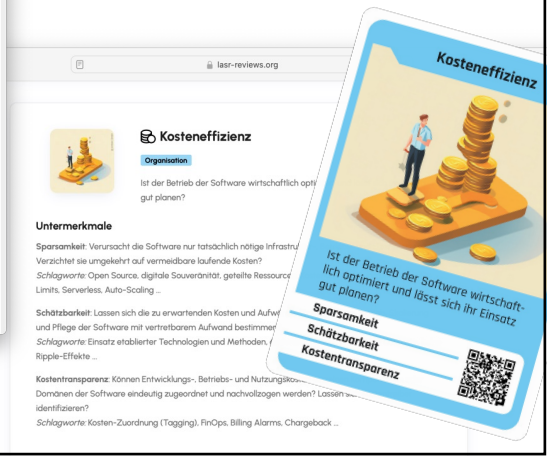


➔ lasr-reviews.org

LASR-Qualitätsmodell

Beschreibungen der in Top-5-Challenger eingesetzten 14 Qualitätsmerkmale online.





➔ lasr-reviews.org/quality-model/

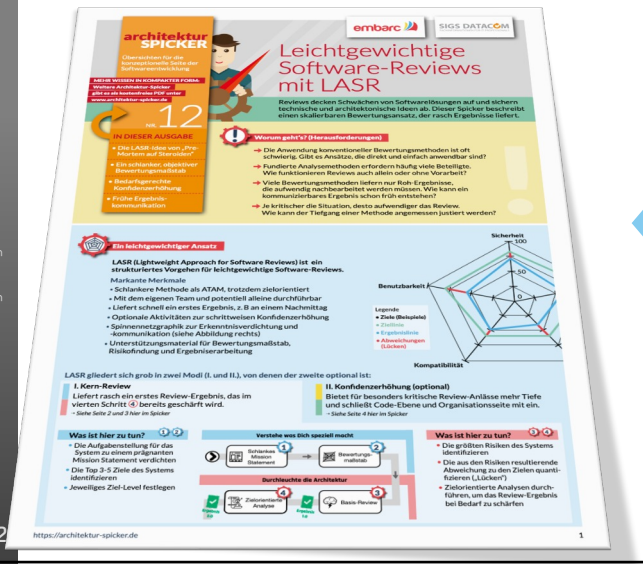
IEEE-Paper

S. Toth, S. Zörner:
„Team-Driven Architecture Evaluation with LASR“
IEEE 23rd International Conference on Software Architecture, Amsterdam 2026

ICSA26 Netherlands · Amsterdam JUNE 22 · 26

➔ lasr-reviews.org/research/lasr-paper-ieee-icsa-2026/

Architektur-Spicker zum Thema …



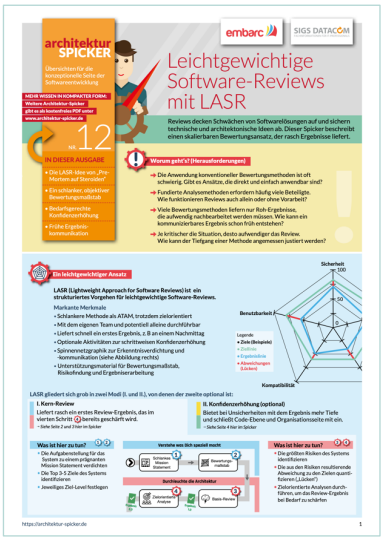
„Mit unseren Architektur-Spickern beleuchten wir die konzeptionelle Seite der Softwareentwicklung."

Architektur-Spicker #12
Leichtgewichtige Software Reviews mit LASR

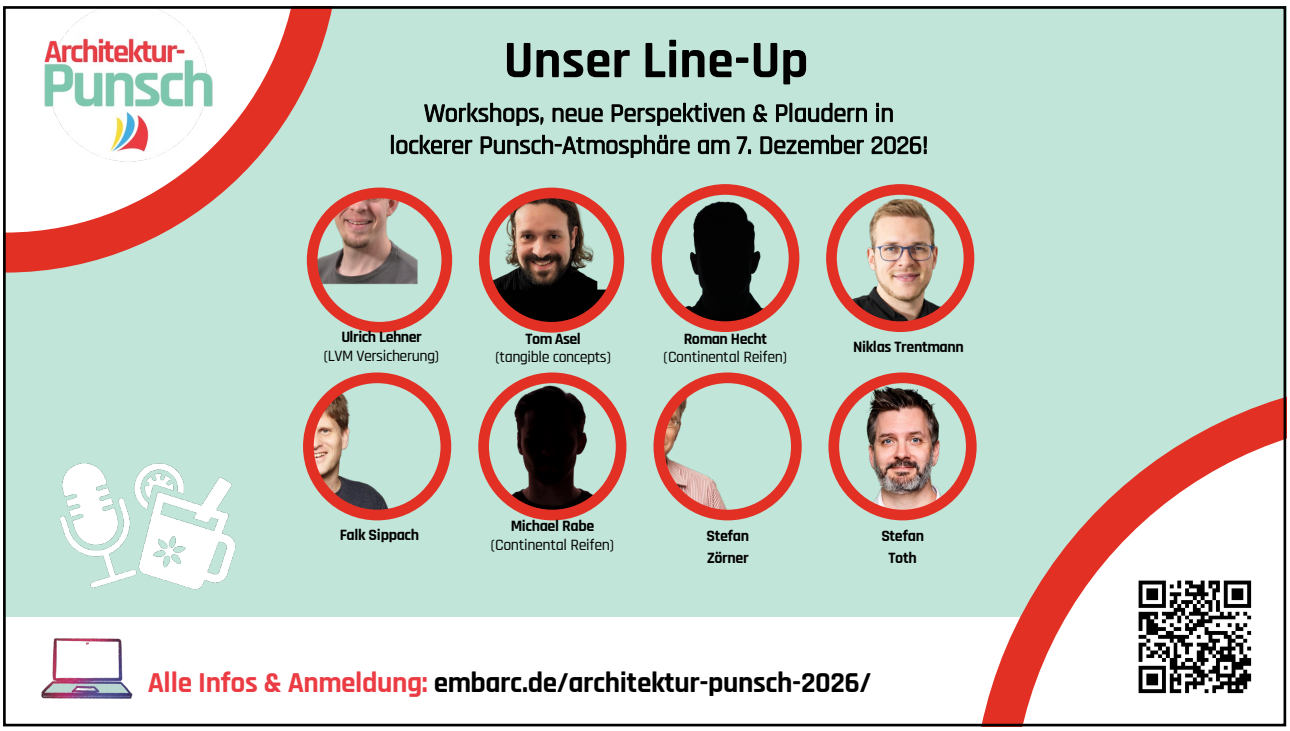
PDF, 4 Seiten
Kostenloser Download.

→ architektur-spicker.de

Zum Mitnehmen



Den LASR-Spicker könnt ihr euch direkt vorne bei mir abholen (solange der Vorrat reicht).


Architektur-
Punsch
Unser Line-Up
Workshops, neue Perspektiven & Plaudern in
lockerer Punsch-Atmosphäre am 7. Dezember 2026!
Ulrich Lehner
(LVM Versicherung)
Tom Asel
(tangible concepts)
Roman Hecht
(Continental Reifen)
Niklas Trentmann
Falk Sippach
Michael Rabe
(Continental Reifen)
Stefan
Zörner
Stefan
Toth
Alle Infos & Anmeldung: embarc.de/architektur-punsch-2026/


embarc.de
Softwarearchitektur in Eigenregie bewerten
95
Vielen Dank.
Gerne Verlinken auf LinkedIn ...
linkedin.com/in/stefan-zoerner
embarc