

Stefan Zörner

Softwarearchitektur im Team in Eigenregie bewerten. Mit LASR an einem Nachmittag

Vortrag bei der JUG Saxony, zu Gast bei Xceptance Software Technologies GmbH
Jena, 10. Juni 2026




embarc.de

Softwarearchitektur in Eigenregie bewerten

1

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 Architektur-Reviews ist es möglich, Schwächen und Risiken von Softwarelösungen herauszuarbeiten, Entscheidungen abzusichern und Verbesserungsmaßnahmen zu bewerten. Klassische Analyseansätze aus diesem Umfeld sind fundiert, kommen aber gerade in beweglichen Softwarevorhaben etwas akademisch und 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 Softwarevorhaben ohne Unterstützung von außen beleuchten und in wenigen Stunden zu ersten, vorzeigbaren 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 Review-Ergebnis.

DO.

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?

01. Warum leichtgewichtig bewerten?
02. Verstehe, was Dich speziell macht
03. Durchleuchte die Architektur
04. Erhöhe die Konfidenz (bei Bedarf)
05. Weitere Informationen




embarc.de

Softwarearchitektur in Eigenregie bewerten

5

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?

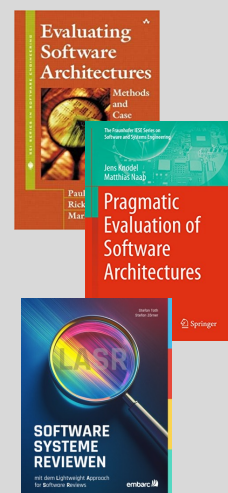


Wörtlich:
Review == etwas **(über)prüfen**, besprechen, ...

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.





embarc.de

Softwarearchitektur in Eigenregie bewerten



Kernelemente eines Reviews

Für eine Bewertung oder ein Review braucht es mindestens



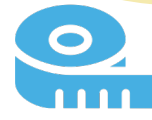
Einen Anlass

Warum bewerten wir?



Einen Gegenstand

Was bewerten wir?



Einen Maßstab

Wonach (wo gegen)
bewerten wir?



embarc.de

Softwarearchitektur in Eigenregie bewerten



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?

Eine Neuentwicklung steht an und erste Lösungsansätze stehen im Raum.



embarc.de

Softwarearchitektur in Eigenregie bewerten

11

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**?

Unterschiedliche Stakeholder verfolgen widersprüchliche Ziele mit eurer Software.



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?



Das Management hat das Vertrauen in eure Lösung verloren.

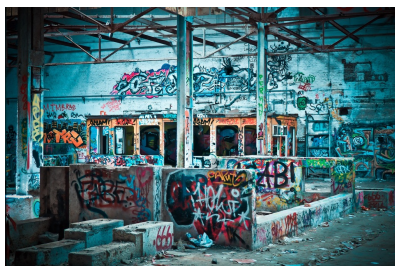


embarc.de

Softwarearchitektur in Eigenregie bewerten

15

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?



Größere Umbaumaßnahmen in eurer Software stehen an.

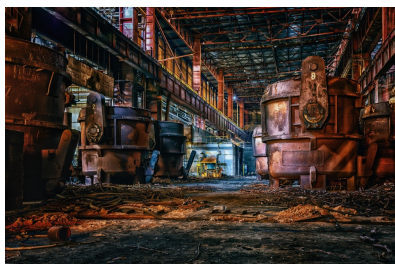


embarc.de

Softwarearchitektur in Eigenregie bewerten

17

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



Anlass 4 („Legacy“)

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

Leitfrage:

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

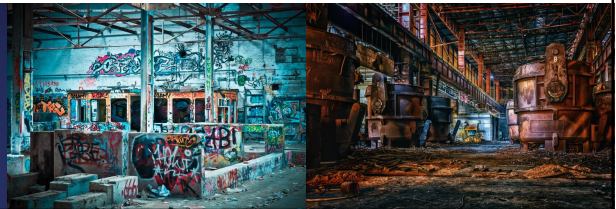


Das Leistungsversprechen von Reviews

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

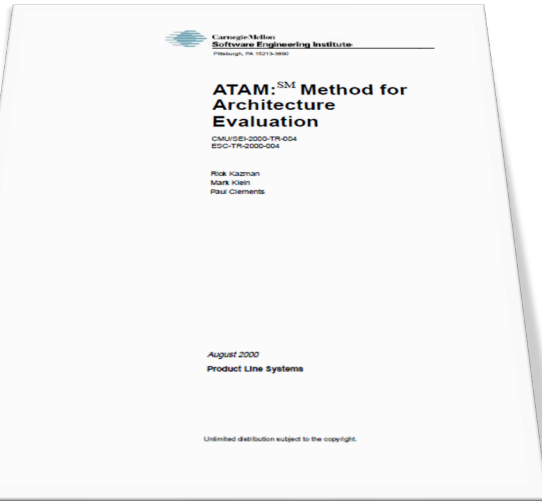


Bewertungsmethoden (Auswahl)

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**.





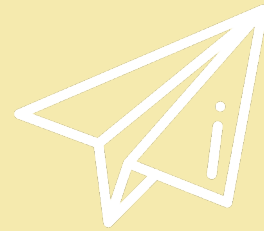
Was ist leichtgewichtig?

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

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



→ leanpub.com/software-systeme-reviewen/

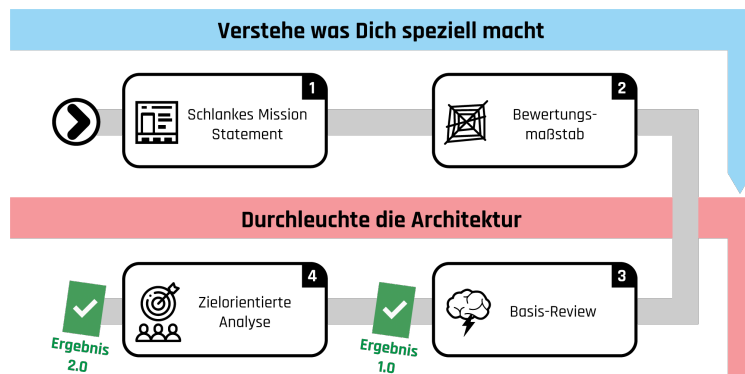


Ein schlanker Ansatz

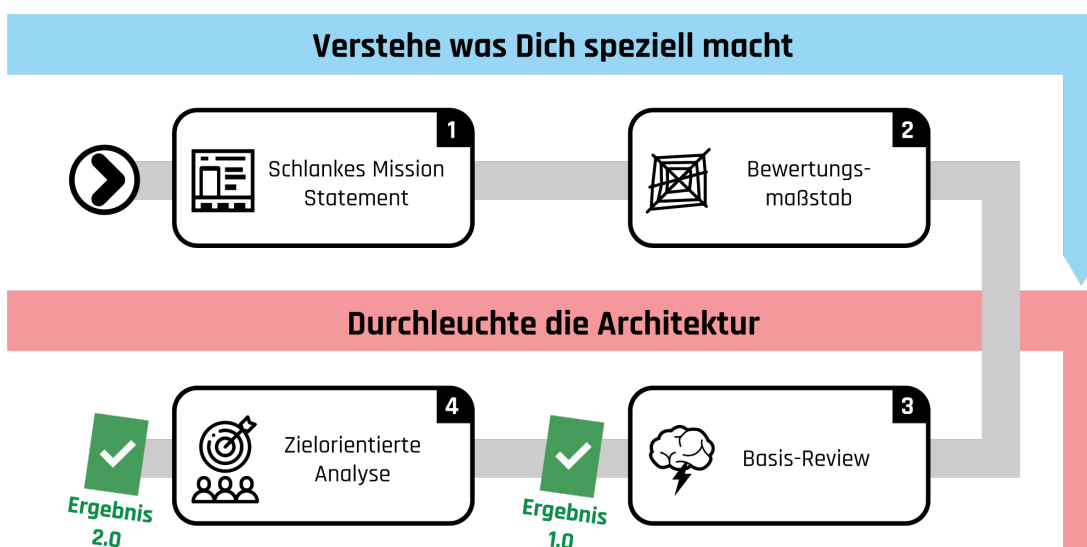
LASR (Lightweight Approach for Software-Reviews) ist ein strukturiertes Vorgehen für leichtgewichtige Software-Reviews.

Ablauf LASR-Review ➔

- 2 Tätigkeiten
- 4 Schritte
- Ein frühes erstes Ergebnis



Tätigkeiten und Schritte im LASR-Review



02.

Verstehe, was Dich speziell macht

01. Warum leichtgewichtig bewerten?

02. Verstehe, was Dich speziell macht

03. Durchleuchte die Architektur

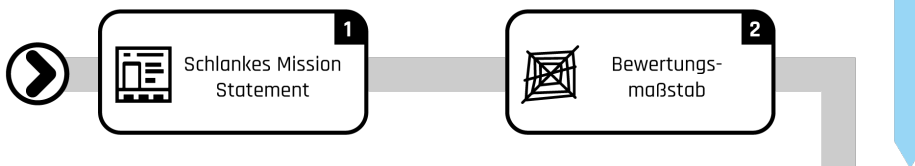
04. Erhöhe die Konfidenz (bei Bedarf)

05. Weitere Informationen



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?



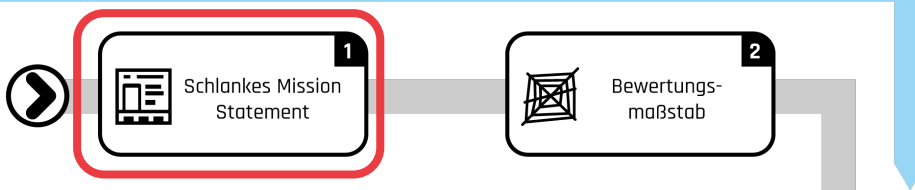
Einen Maßstab

Wonach (wo gegen)
bewerten wir?



Verstehe, was Dich speziell macht

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

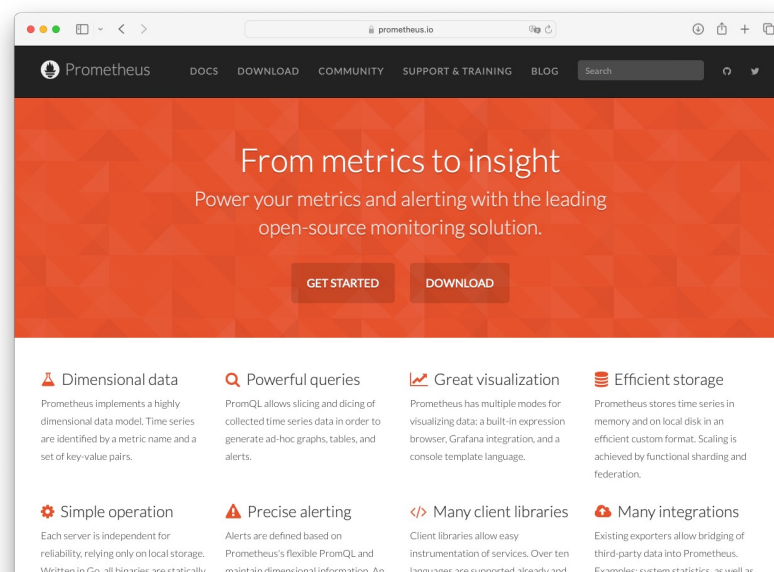
Sammeln der Claim(s) mit **Landing Page-Metapher**

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

Erarbeiten im Workshop

LASR Schritt 1: Schlankes Mission Statement

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

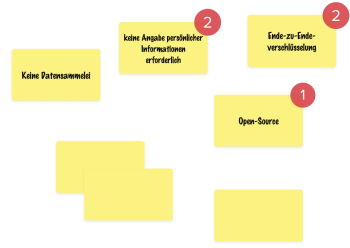


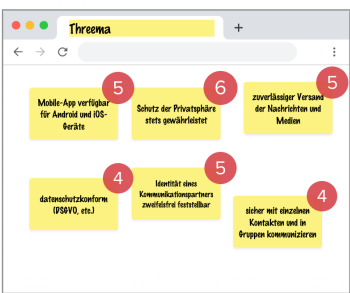
1 Brainstorming

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.





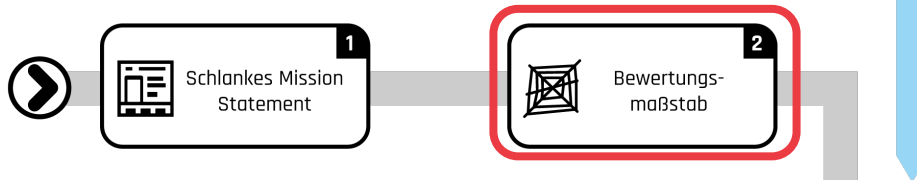
Quelle: S. Toth, S. Zörner: „Software-Systeme reviewen“, Leanpub 2023

Beispiel im digitalen Whiteboard (Mural)



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



(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



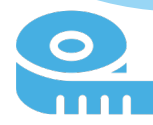
Einen Anlass

Warum bewerten wir?



Einen Gegenstand

Was bewerten wir?

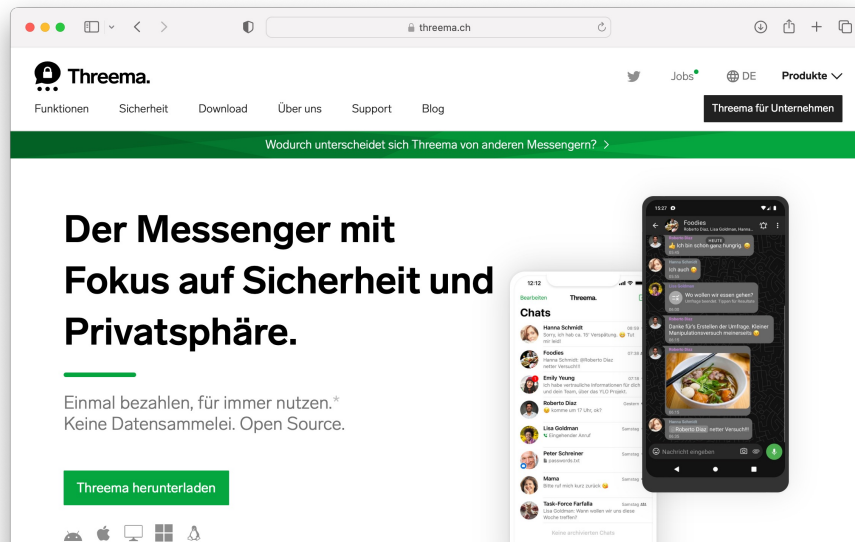


Einen Maßstab

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





embarc.de

Softwarearchitektur in Eigenregie bewerten

43

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.



Verfügbarkeit



Sicherheit



Performanz



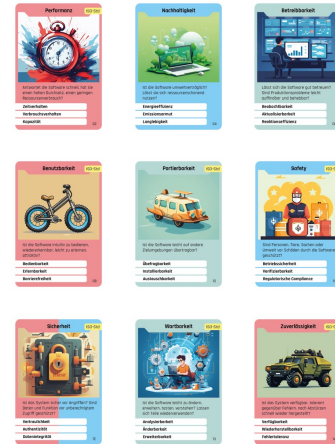
Wartbarkeit



Top-5-Challenger, Vorbereitung (Beispiel)



Start Top-5



Kandidaten



Ablauf einer Runde



Einen „Challenger“ nominieren

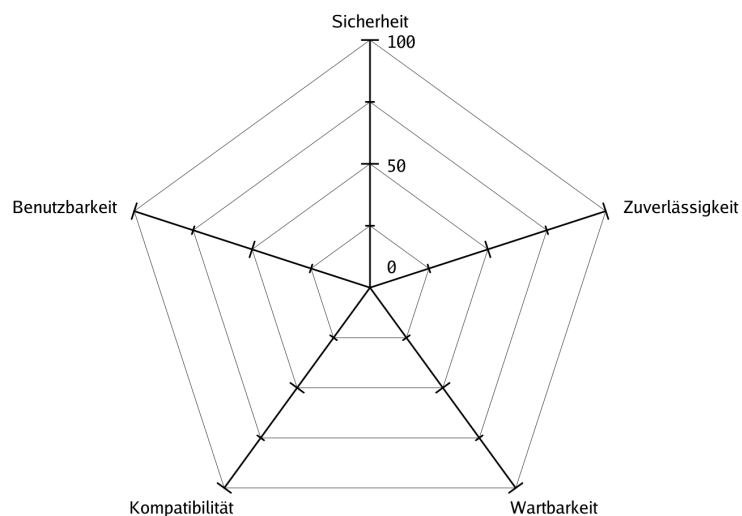
- Team-Mitglied schlägt Ziel vor, das in Top-5 fehlt.
- Die Gruppe diskutiert darüber.
- Der Challenger verdrängt eine Karte oder landet selbst auf dem Ablagestapel.



Nach 2 Runden (Beispiel)

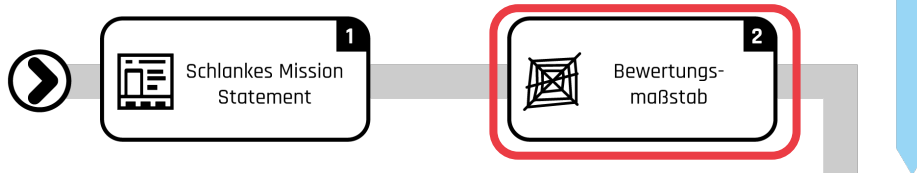


Ergebnis nach diesem Schritt (Beispiel)



Verstehe, was Dich speziell macht

Verstehe was Dich speziell macht



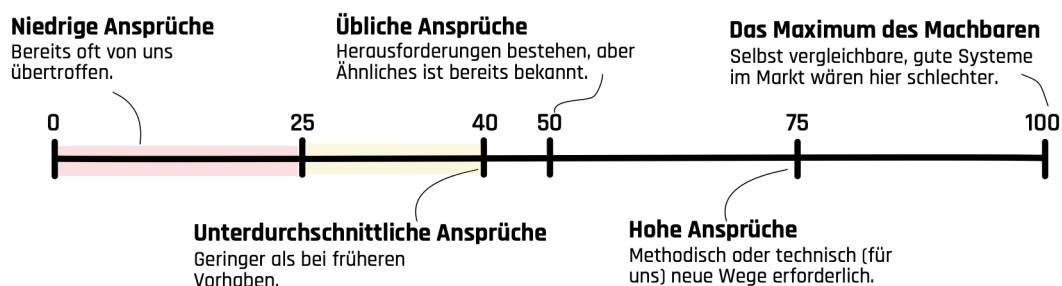
Was ist zu tun?

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

48

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.

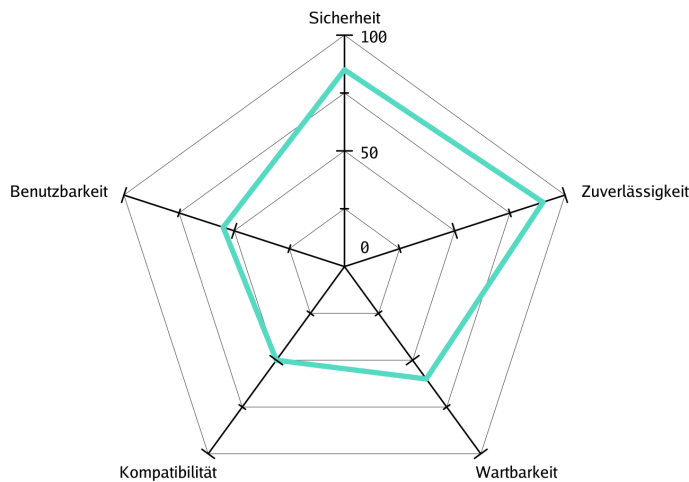


Hinweis: Es handelt sich hier um eine Einschätzung, nichts Messbares.

49



Bewertungsmaßstab einzeichnen



LASR-Ergebnisdiagramm

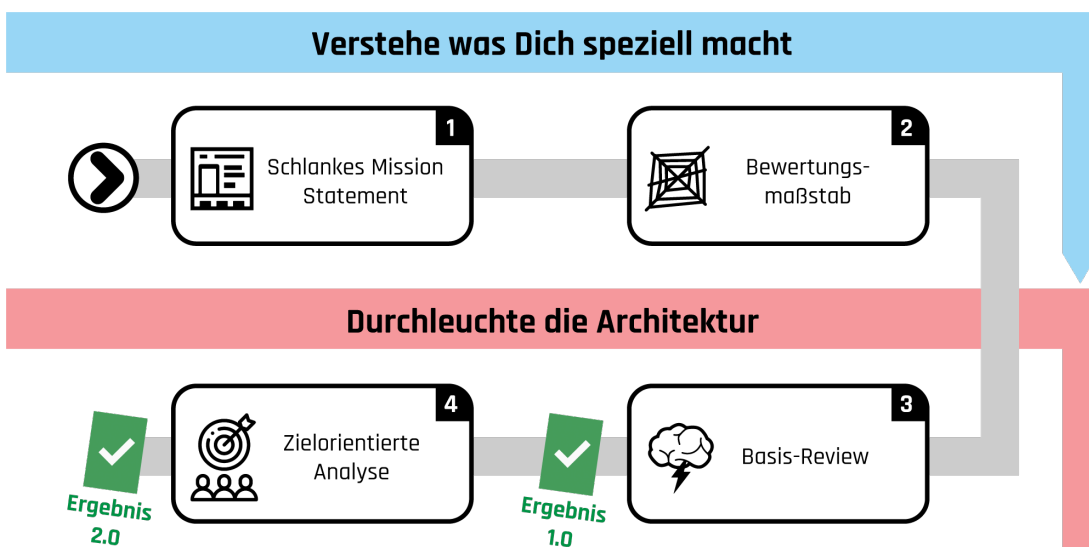
Die Top-3-5 Qualitätsziele bilden die Achsen des Diagramms.

Die Zielwerte spannen darauf eine grüne Linie auf.

← Beispiel



Mit Schritt 2 steht der Ergebnisrahmen.



03.

Durchleuchte die Architektur

01. Warum leichtgewichtig bewerten?
02. Verstehe, was Dich speziell macht
- 03. Durchleuchte die Architektur**
04. Erhöhe die Konfidenz (bei Bedarf)
05. Weitere Informationen





embarc.de

Softwarearchitektur in Eigenregie bewerten

53

Durchleuchte die Architektur

Durchleuchte die Architektur



Ergebnis
2.0

4



Zielorientierte Analyse



Ergebnis
1.0

3



Basis-Review

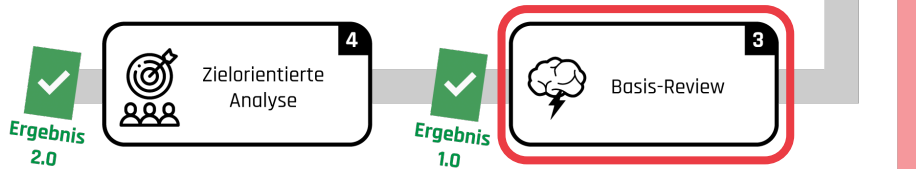
Was ist zu tun?

- 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

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**

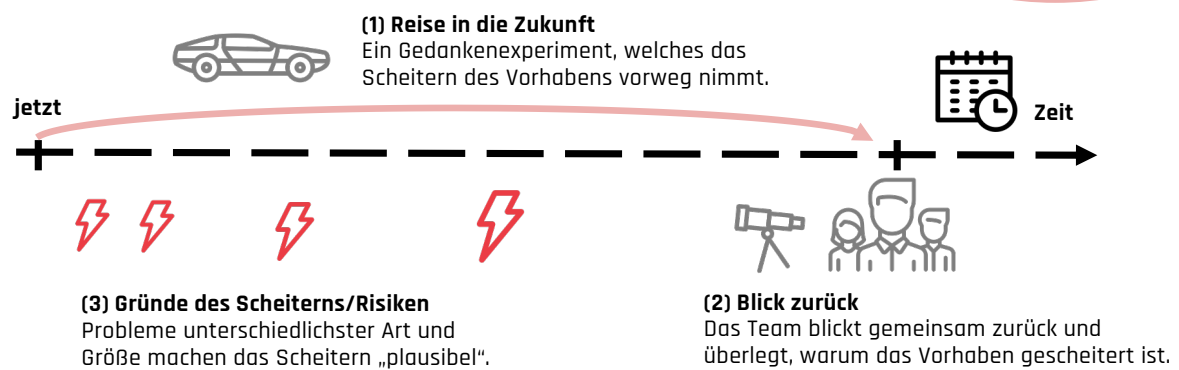
Ergebnisse des Schrittes

"Abweichung" vom Ziel, Ist-Einschätzung

Grundidee: Pre Mortem



Was ist ein Pre-Mortem?





Die 8 Risiko-Kategorien von LASR



Softwarelösung 1

- Unpassende Technologien
- Komplexe Lösungen
- Unzureichende Fremdsysteme
- 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/Make 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
- Einengende Standards
- Isolierte Entscheidungskompetenzen
- ...



Betrieb und Deployment 7

- Blockierende Prozesse
- Geringe Automatisierung
- Wenig Feedback
- Fehlender Betrieb / Ausfallkonzepte
- Tool-Probleme
- ...



Weiche Faktoren 8

- Uneingetriggten
- Konflikte
- Fehlende Disziplin
- Kommunikationsbarrieren
- Rollenethiken
- Unpassende Kultur
- ...



Typische Risikothemen als Ideengeber



Zu hohe Komplexität der Lösung

Hat die Domäne eine sehr hohe Komplexität? Gibt es überlegte, 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 Abfrage, 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?

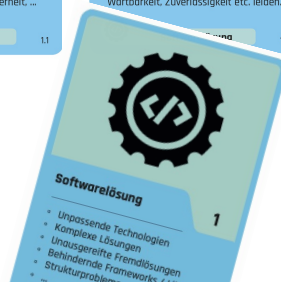
Softwarelösung 1.3



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



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



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?



Zielsetzung



Hypothetisches Risiko (1)



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

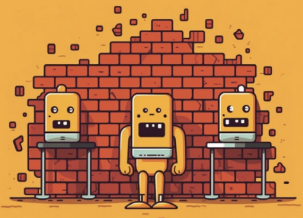
Aufhebung des Gruppen-Limits

„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)

Vendor-Lock-In oder Support-Probleme

Gibt es Abhängigkeiten von Lieferanten, deren Ziele oder Einsatzzwecke abweichen? Stehen Abkündigungen im Raum? Gibt es potentielle Probleme bei Support, Lizenzmodellen, Kosten, ...?

 **Fremdsysteme & Plattformen** 4.4

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)




Einengende Standards oder Randbedingungen

Werden Architekturentscheidungen (oft) durch Standards, Budget- oder Zeitbeschränkungen erschwert? Gibt es einschneidende rechtliche Aspekte rund um Daten oder Internationalisierung?

 **Organisation & Prozesse** 6.4

Private Kommunikation scannen

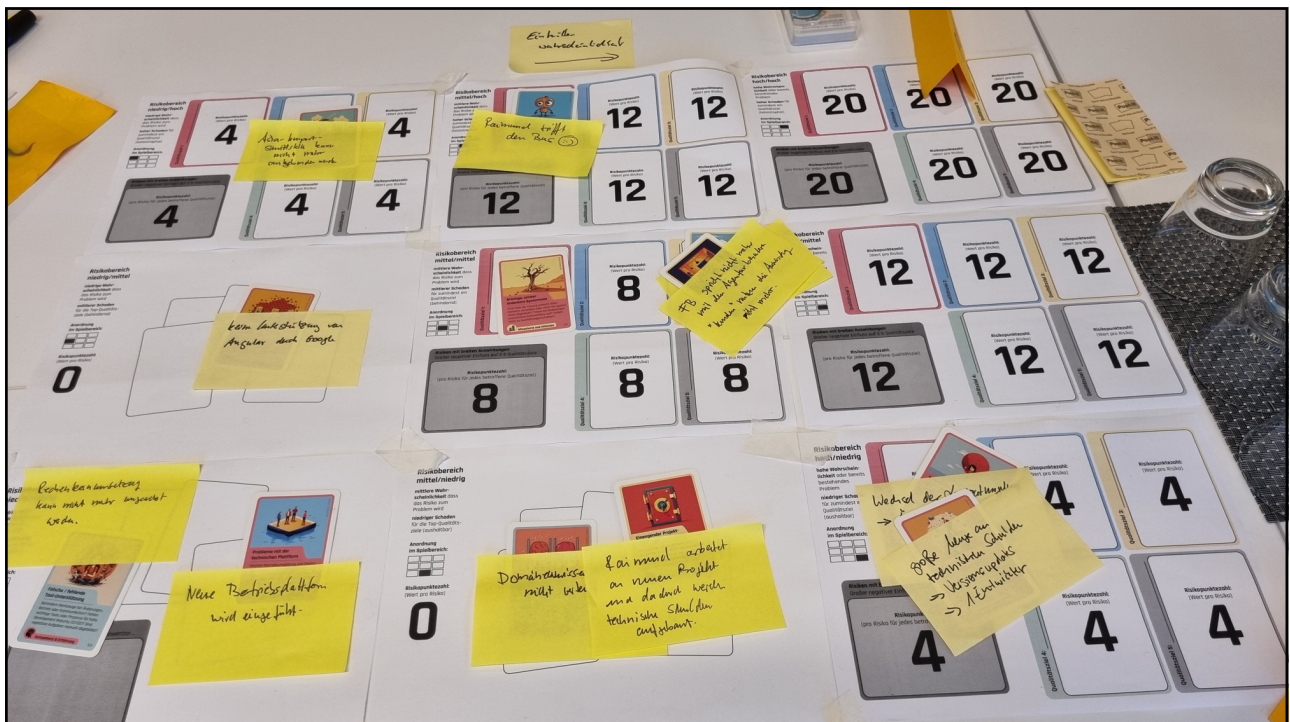
„Die europäische Gesetzgebung hat Messenger-Anbieter verpflichtet, sämtliche private Kommunikation und Dateien zu durchleuchten. Threema muss daraufhin die Lösung für private Nutzer komplett neu denken.“



Risiken einordnen

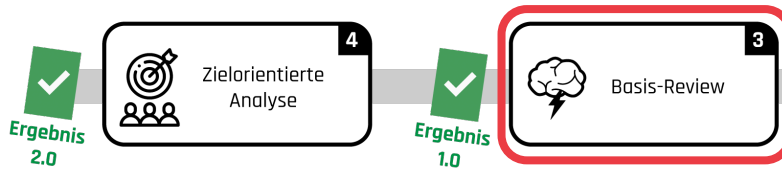


Schadenshöhe	katastrophal	4	12	20
	behindernd	0	8	12
	aushaltbar	0	0	4
		niedrig	mittel	hoch
		Eintrittswahrscheinlichkeit		



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

Von Risiken zu „Lücken“



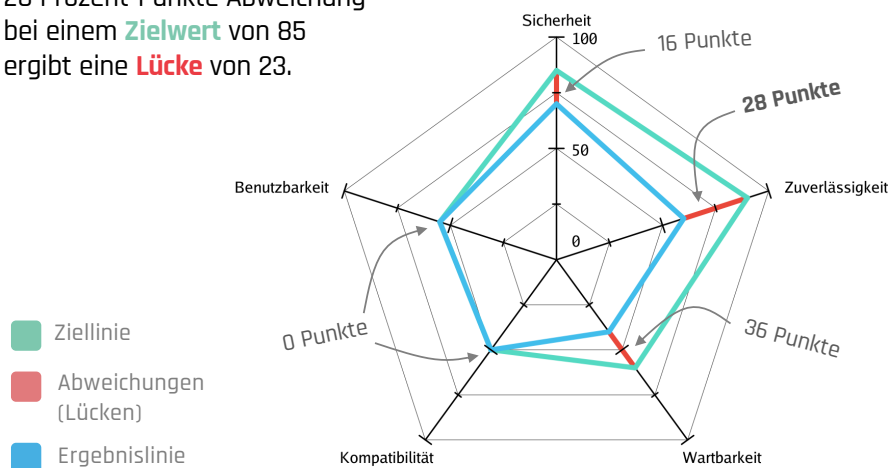
28% Abweichung
vom Ziellevel
auf der Qualitätszielachse



Abweichungen einzeichnen

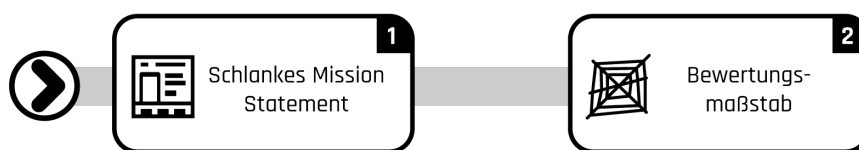
Beispiel

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



Basis-Review, 1. Ergebnis nach Schritt 3

Verstehe was Dich speziell macht



Durchleuchte die Architektur





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 anklängen lässt - ggf. als Set von „Claims“ aus einem Brainstorming.]

Claim

Claim

Claim

Claim

LASR Ergebnisdigramm (Spinnennetzgrafik)

A spider diagram with five axes: Performance (0-100), Zuverlässigkeit, Benutzbarkeit, Portierbarkeit, and Wartbarkeit. A blue line connects points on these axes, with red numbers 1, 2, 3, 4, 5 marking specific points.

Lücken und Risiken

1 [Kurze Lücken-Beschreibung + ausgewählte Details]

Risiko (2C) 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

Begrüßung, Einstieg LASR

ca. 10 Minuten

1

Abgrenzung, Mission Statement ("Claims")

ca. 15-20 Minuten

2

Bewertungsmaßstab festlegen

ca. 20-30 Minuten

kurze Pause (+ Basis-Review vorbereiten)

10:15 - 11:30

3

Basis-Review ("Pre-Mortem")

ca. 60 Minuten

3

Zielerreichung betimmen

ca. 15 Minuten

kurze Pause

11:45 - 12:15

3

Ergebnisse diskutieren

ca. 10 Minuten

3

Ziel-orientierte Analyse planen, TODOs

ca. 10-20 Minuten

Ende des Workshops

← Beispiel-Agenda

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

(c) embarc GmbH

38

04.

Erhöhe die Konfidenz (bei Bedarf)

01. Warum leichtgewichtig bewerten?

02. Verstehe, was Dich speziell macht

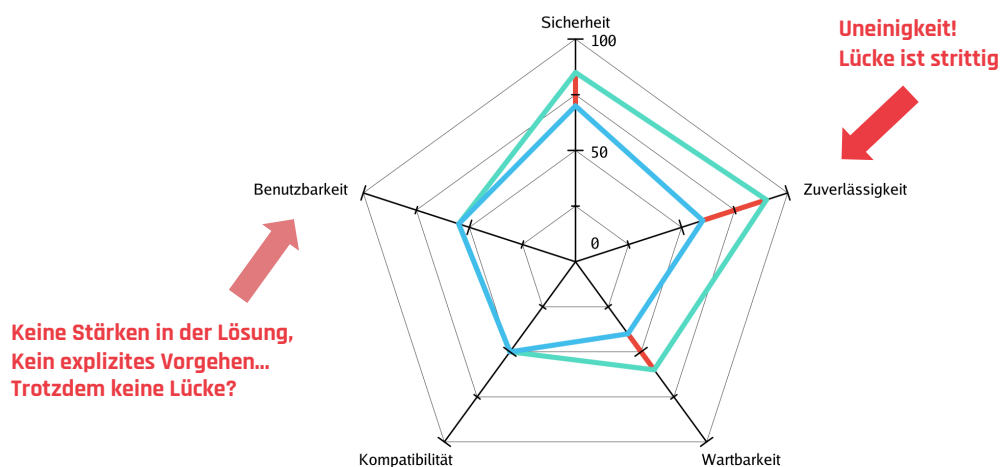
03. Durchleuchte die Architektur

04. Erhöhe die Konfidenz (bei Bedarf)

05. Weitere Informationen



Fokus: Zielorientierte Analyse (Beispiele)



embarc.de

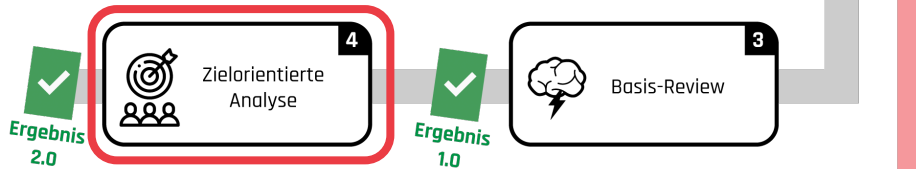
Softwarearchitektur in Eigenregie bewerten

77



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**



(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 →

Zielachse

Lücke (Schritt 3):

36

Lücke geschätzt:

20

Zuverlässigkeit (100-0%)

ist das System verfügbar, liefert gegenüber Fehler, nach Ausfällen schnell wieder reagiert?

Verfügbarkeit

Wiederherstellbarkeit

Fehlertoleranz

Konkrete Risiken

Welche Risiken sind der Zielachse zugeordnet?

1. Risiko: Datenverlust durch Hardwareausfall
keine QA betroffen
keine Mäßigung

2. Risiko: Veränderung der Anforderungen
keine QA betroffen
keine Mäßigung

3. Risiko: Ausfall der Netzwerke
keine QA betroffen
keine Mäßigung

4. Risiko: Erkennung von Ausfall und Ursache ist manuell
keine QA betroffen
keine Mäßigung

5. Risiko: Risiko-Einschätzung

1C	2C	3C
1B	2B	3B
1A	2A	3A

Entwicklungsphase

Qualitätsaussagen

Welche Aussagen illustrieren die Zielachse?

1. Kernkomponenten werden auf der Basis von...
H
HMM
L
HMM

2. Die Verwendung von...
H
HMM
L
HMM

3. Es kommt zu einem...
H
HMM
L
HMM

4. Zentrale QAs - Qualitätsaussagen

5. Technisches Risiko dass QA erreicht wird: High / Medium / Low

Stärken der Lösung

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

1. Message-basierte Kommunikation (asynch. nstng-fähig)

2. Produktion einer dedizierten Verbindung zwischen Komponenten (asynch. nstng-fähig)

3. Vermeidung der VCS auch bei synchroner + asynchroner Kommunikation

4. Verteilungsskizze kann 3D ohne Schnitt überprüften

Schwächen der Lösung

Welche Lösungsaspekte wirken problematisch?
Welche Aspekte verschulden Risiken?

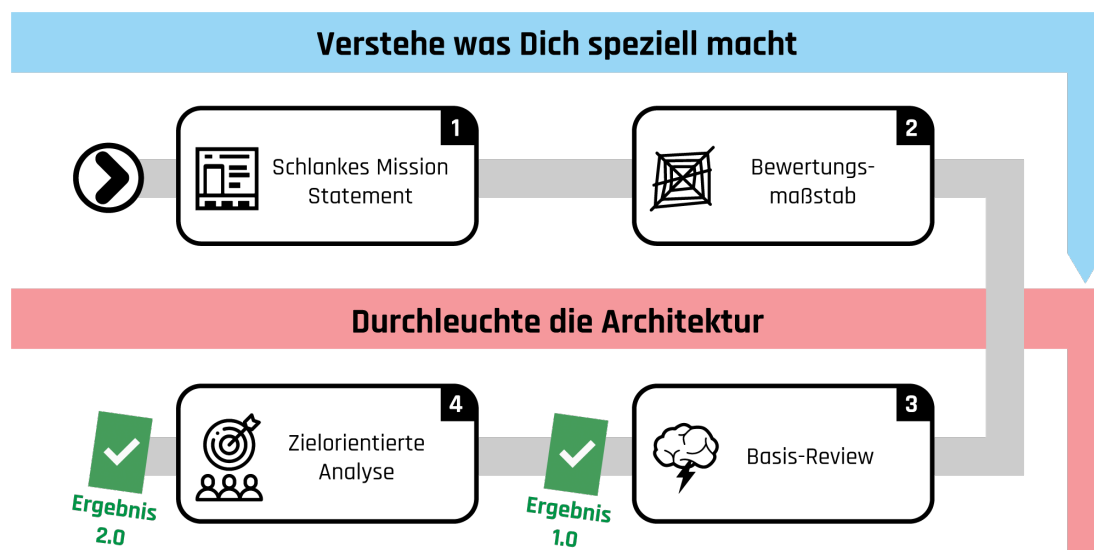
1. Fehlererkennung über Videos konstant viel Bandbreite

2. Noch keine Erkennung von Verbindungsproblemen

3. Datenintensive Verbindung über VPN



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?

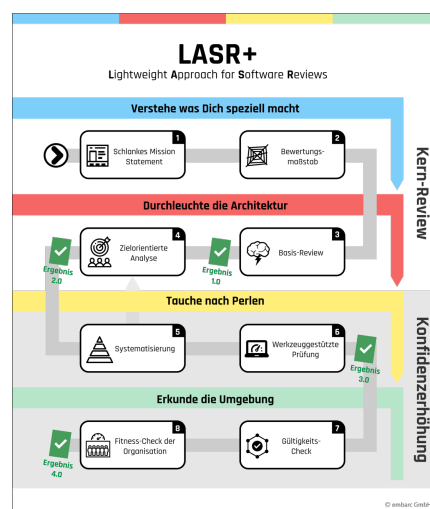
Die **Priorisierung** der identifizierten Probleme ist **schwierig**, ggf. fehlt es bei kleinteiligen Ergebnissen an Überblick oder Einordnung.

Ist nicht auch XY wichtig?

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



Ausblick: Nach dem LASR-Review ...



LASR bietet einen erweiterten, optionalen Modus (LASR+) zur Konfidenzerhöhung.

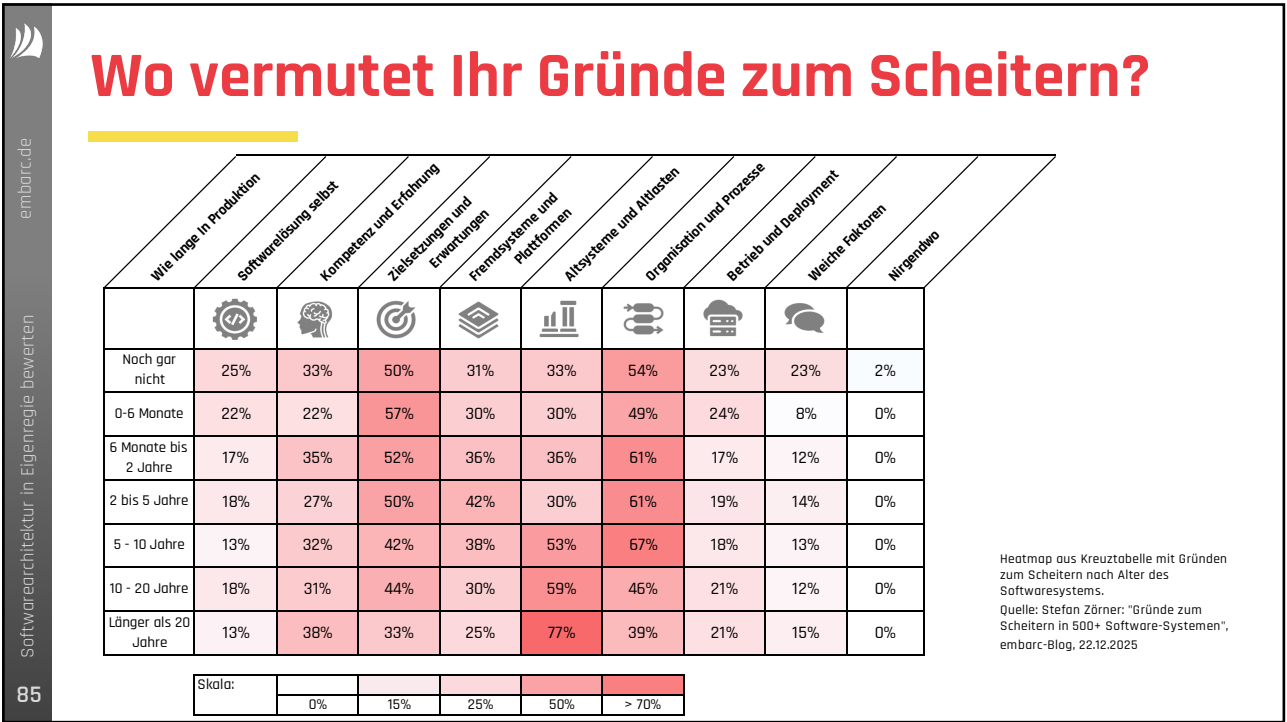
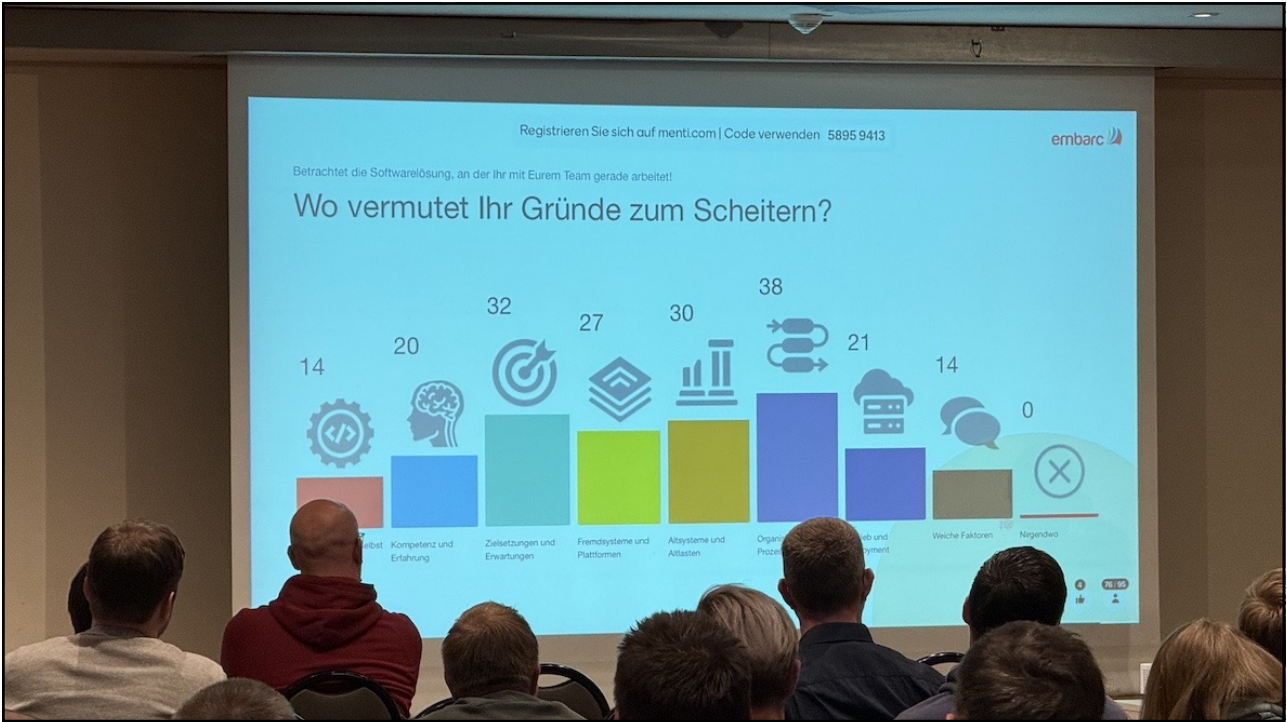
I. LASR-Review

Liefert rasch ein erstes Review-Ergebnis, das im vierten Schritt bereits geschärft wird.

II. LASR+ (optional)

Bietet bei Unsicherheiten mit dem Ergebnis mehr Tiefe und schließt Code-Ebene und Organisationsseite mit ein.





05.

Weitere Informationen

01. Warum leichtgewichtig bewerten?

02. Verstehe, was Dich speziell macht

03. Durchleuchte die Architektur

04. Erhöhe die Konfidenz (bei Bedarf)

05. Weitere Informationen



embarc.de

Softwarearchitektur in Eigenregie bewerten

87

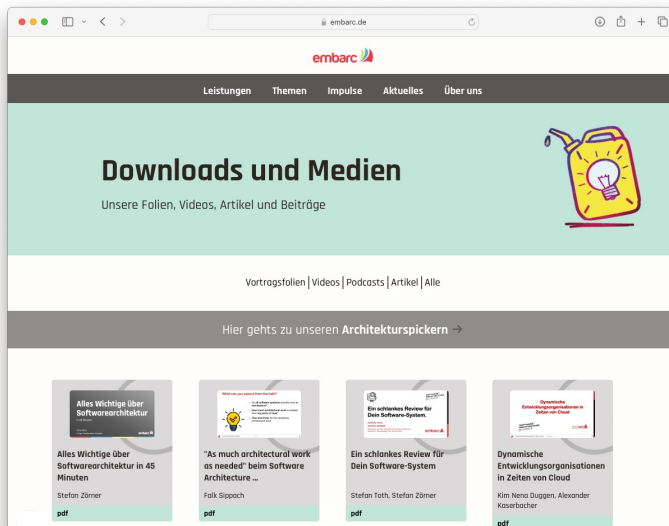
Markante Merkmale von LASR

- 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

Autoren: Stefan Toth, Stefan Zörner

Verlag: Leanpub, September 2023

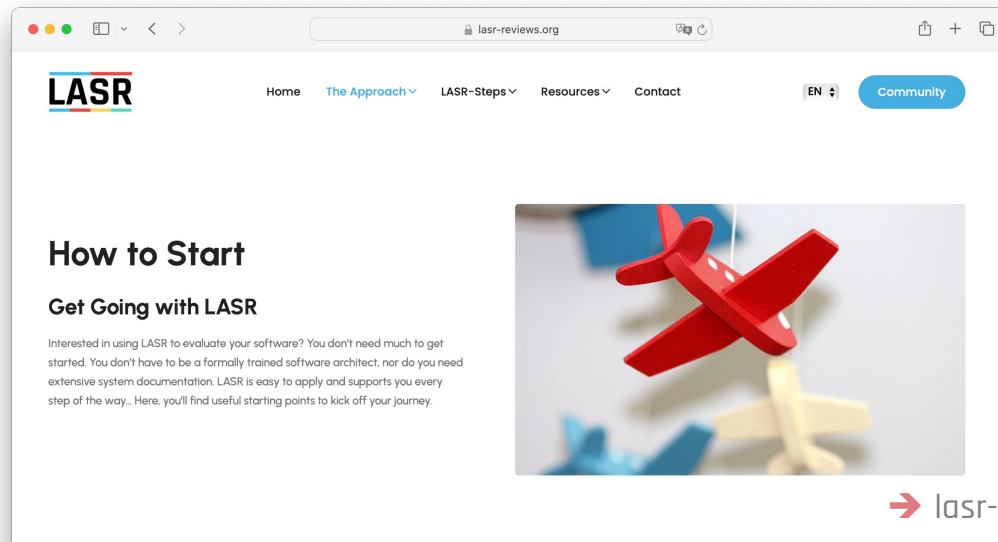
Sprache: Deutsch, EPUB, PDF



→ leanpub.com/software-systeme-reviewen/



„Offizielle“ LASR-Webseite (EN + DE)

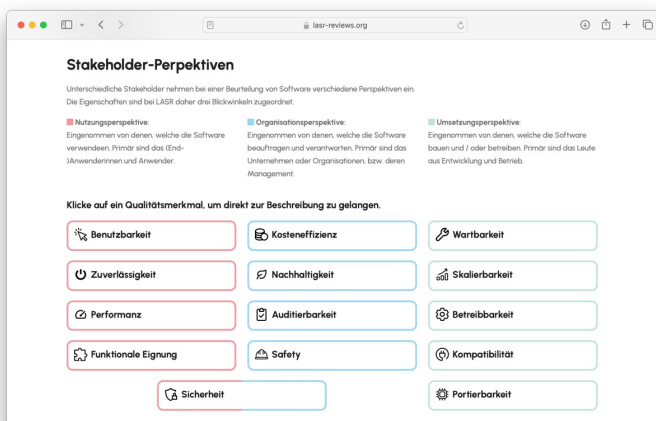


→ lasr-reviews.org

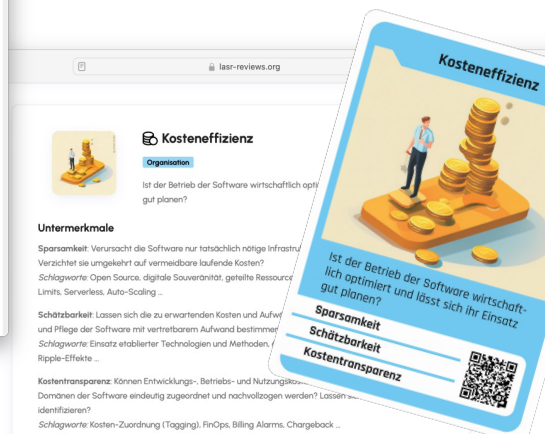


LASR-Qualitätsmodell

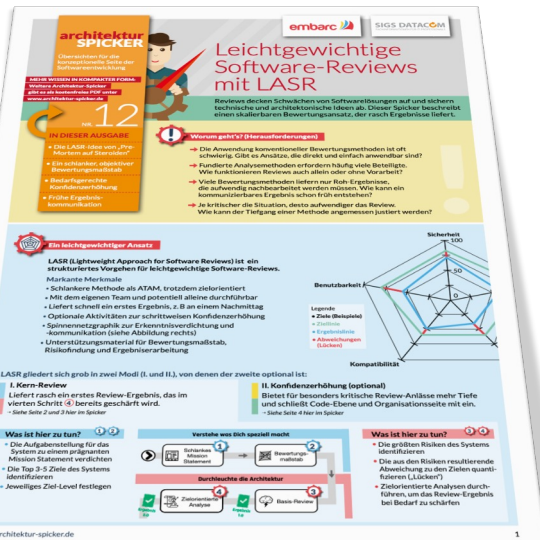
Beschreibungen der in Top-5-Challenger eingesetzten 14 Qualitätsmerkmale online.



→ lasr-reviews.org/quality-model/



Architektur-Spicker zum Thema ...



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

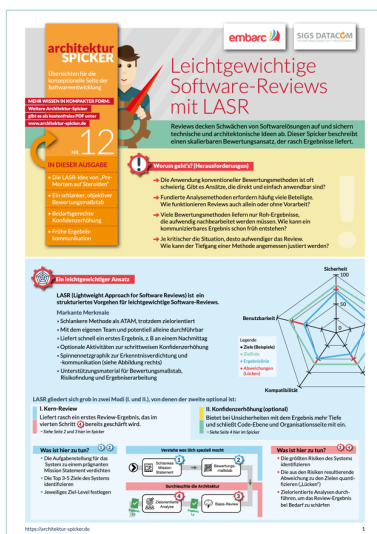
Architektur-Spicker #12
Leichtgewichtige Software
Reviews mit LASR



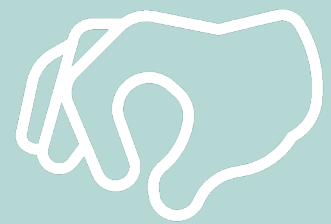
PDF, 4 Seiten
Kostenloser Download.

➔ architektur-spicker.de

Zum Mitnehmen



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





Unser Line-Up

Vorträge und Workshops, neue Perspektiven & Plaudern
in lockerer Punsch-Atmosphäre am 7. Dezember 2026!



Ulrich Lehner
(LVM)



Tom Asel



Roman Hecht
(Continental Reifen)



Niklas Trentmann



Falk
Sippach



Michael Rabe
(Continental
Reifen)



Stefan
Zörner



Stefan
Toth




Alle Infos & Anmeldung: embarc.de/architektur-punsch-2026/



Vielen Dank.

Gerne Verlinken auf LinkedIn ...

 linkedin.com/in/stefan-zoerner



