

Stefan Zörner



Die Softwarearchitektur eures Systems in Eigenregie bewerten

Vortrag bei der Softwerkskammer
PRISMA European Capacity Platform GmbH
Leipzig, 24. September 2025



Stefan Zörner

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

Schwerpunkte

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





Abstract

embarc.de

Softwarearchitektur in Eigenregie bewerten

2

Die Softwarearchitektur eures Systems in Eigenregie bewerten.

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.

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



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

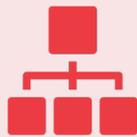
embarc.de

Softwarearchitektur in Eigenregie bewerten

6

Zerlegung

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



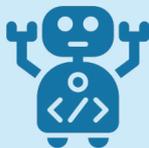
Zielumgebung

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?

embarc.de

Softwarearchitektur in Eigenregie bewerten

7



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



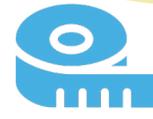
Einen Anlass

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?)

embarc.de

Softwarearchitektur in Eigenregie bewerten

10



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?)

embarc.de

Softwarearchitektur in Eigenregie bewerten

11



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?)

embarc.de

Softwarearchitektur in Eigenregie bewerten

12



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?)

embarc.de

Softwarearchitektur in Eigenregie bewerten

13



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)

embarc.de

Softwarearchitektur in Eigenregie bewerten

16

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)

embarc.de

Softwarearchitektur in Eigenregie bewerten

17

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

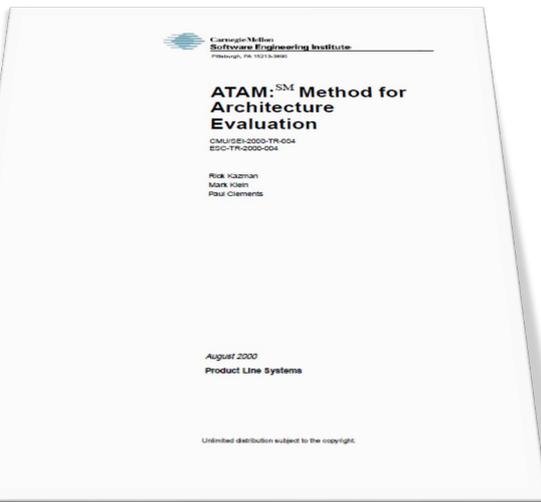


ATAM

embarc.de

Softwarearchitektur in Eigenregie bewerten

10



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 ...

embarc.de

Softwarearchitektur in Eigenregie bewerten

19

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**.





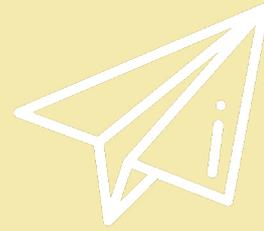
Was ist leichtgewichtig?

Merkmale eines leichtgewichtigen Reviews

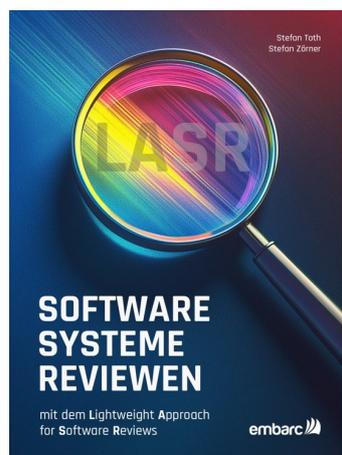
- Die **Anzahl** der Beteiligten / **Stakeholder** ist **gering**.
- **Aufwand** und Dauer sind **überschaubar**.
- **Ergebnisse** liegen vergleichsweise **schnell** vor.

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/

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

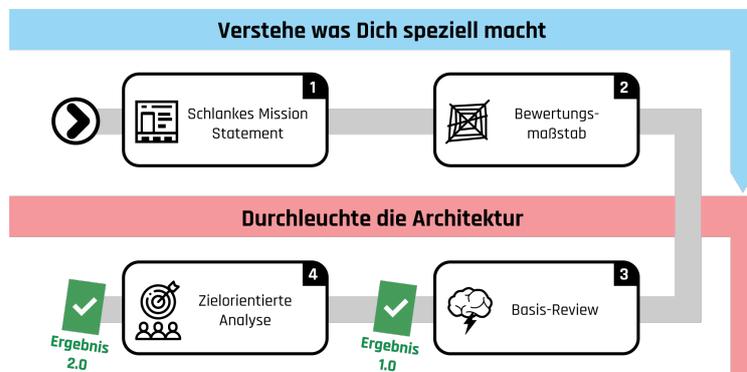


Ein schlanker Ansatz

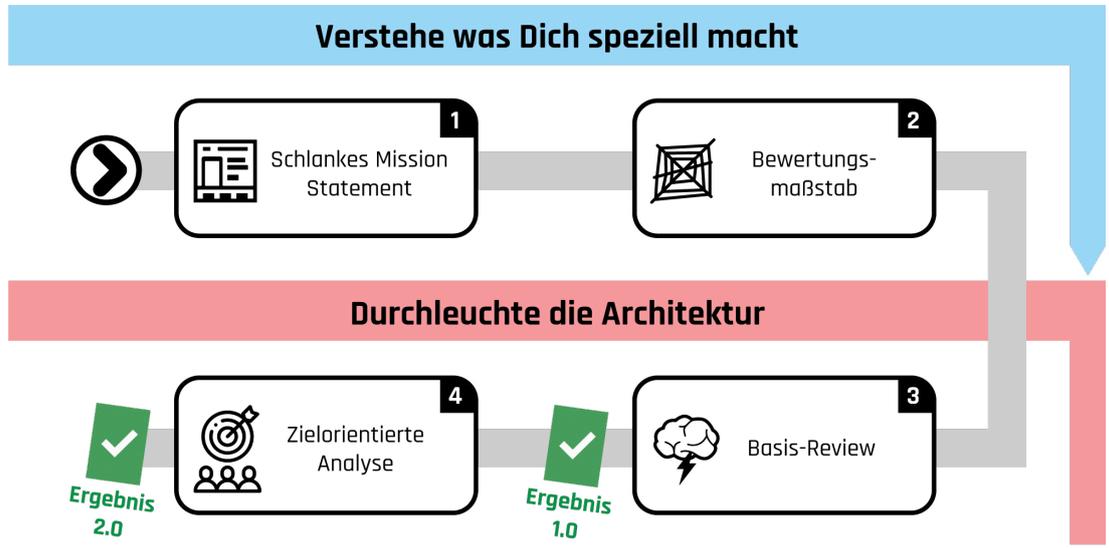
LASR (Lightweight Approach for Software-Reviews) ist ein strukturiertes Vorgehen für leichtgewichtige Software-Reviews.

Ablauf LASR-Review →

- 2 Tätigkeiten
- 4 Schritte
- Ein frühes erstes Ergebnis



Tätigkeiten und Schritte im LASR-Review

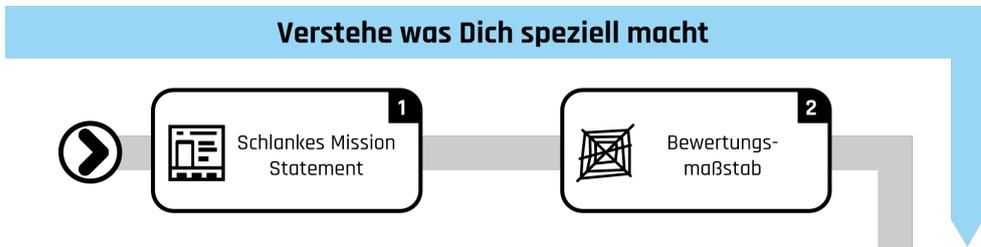


02.

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



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?

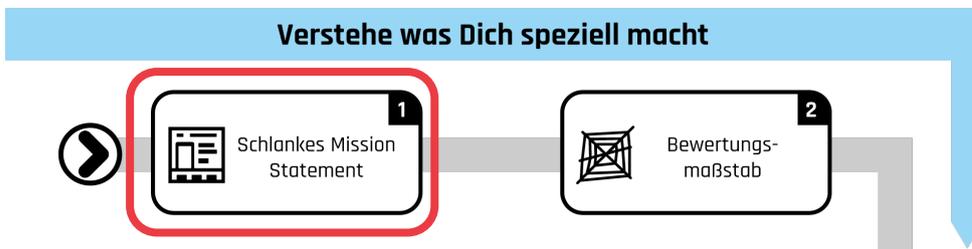


Verstehe, was Dich speziell macht

embarc.de

Softwarearchitektur in Eigenregie bewerten

28



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

embarc.de

Softwarearchitektur in Eigenregie bewerten

29



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

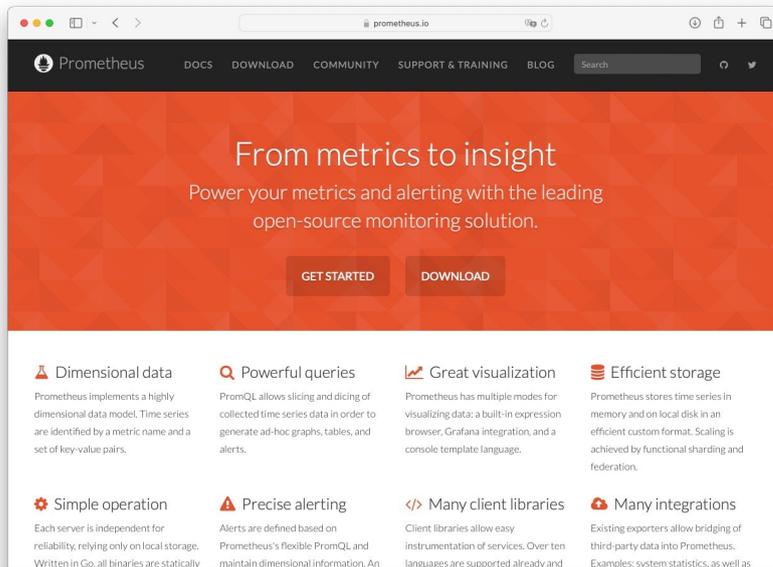
Methodischer Ansatz

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

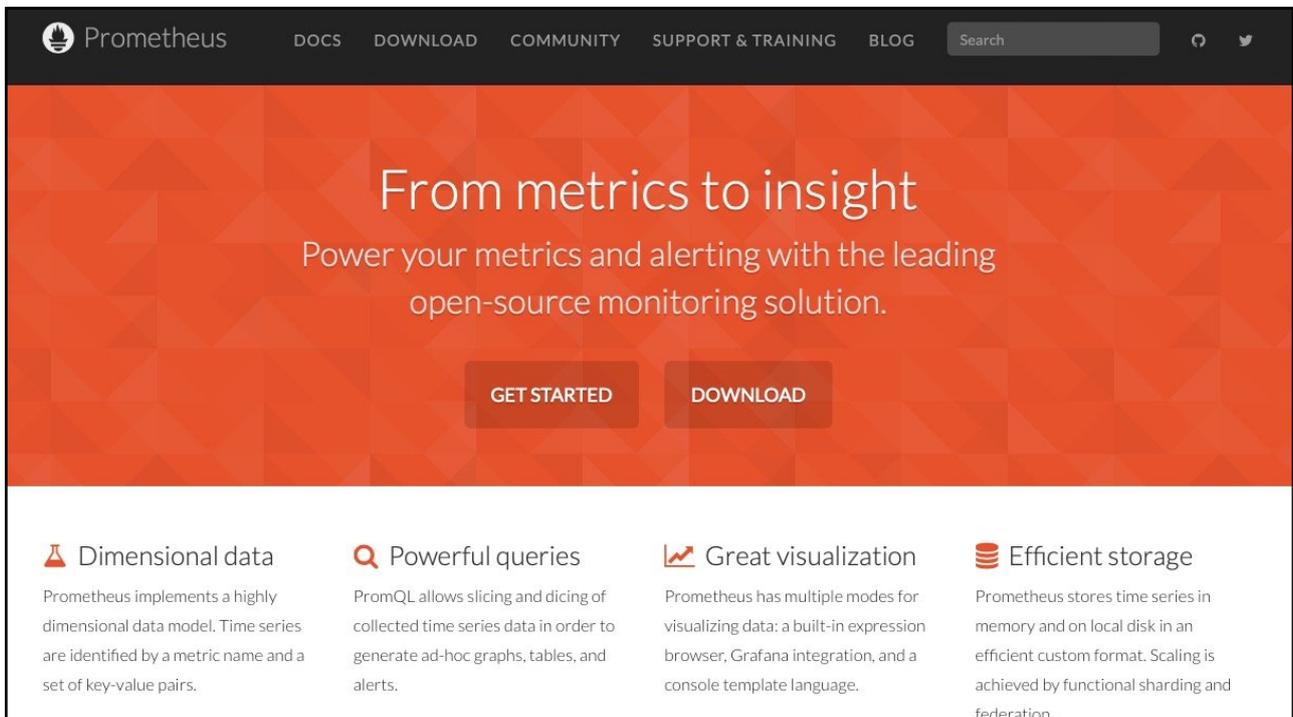
Ergebnis des Schrittes

Prägnante Produktidee (Stichpunkte)

Beispiel: Landing Page von Prometheus



[→ prometheus.io](https://prometheus.io)



Erarbeiten im Workshop

embarc.de

Softwarearchitektur in Eigenregie bewerten

32

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)

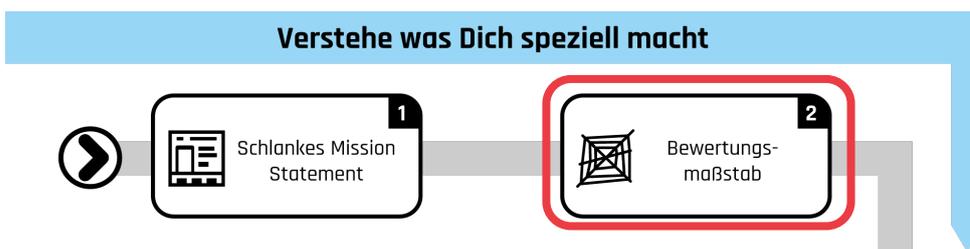
Quelle: S. Toth, S. Zörner: „Software-Systeme reviewen“, Leanpub 2023

Verstehe, was Dich speziell macht

embarc.de

Softwarearchitektur in Eigenregie bewerten

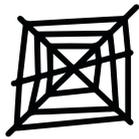
33



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



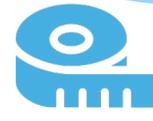
Einen Anlass

Warum bewerten wir?



Einen Gegenstand

Was bewerten wir?

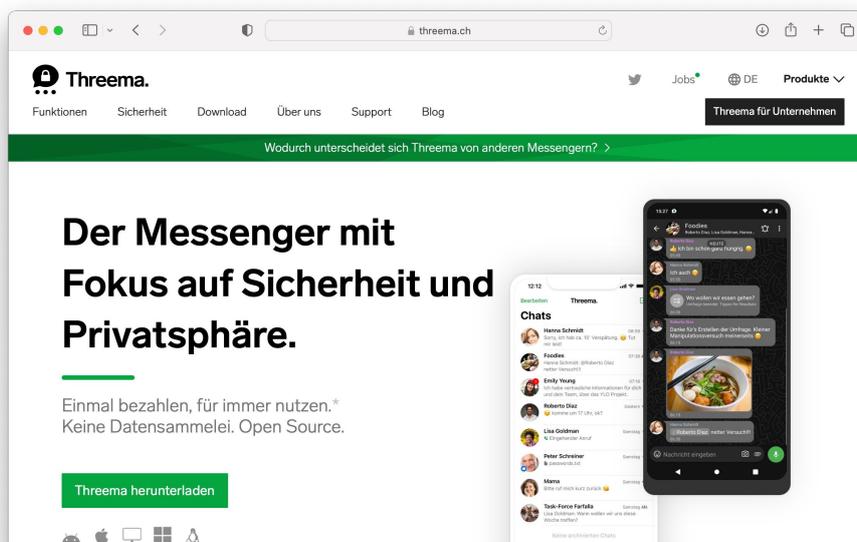


Einen Maßstab

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.



embarc.de

Softwarearchitektur in Eigenregie bewerten

42

Nach 2 Runden, Beispiel (Beispiel)

Aktuelle Top-5

Ablagestapel

Kandidaten



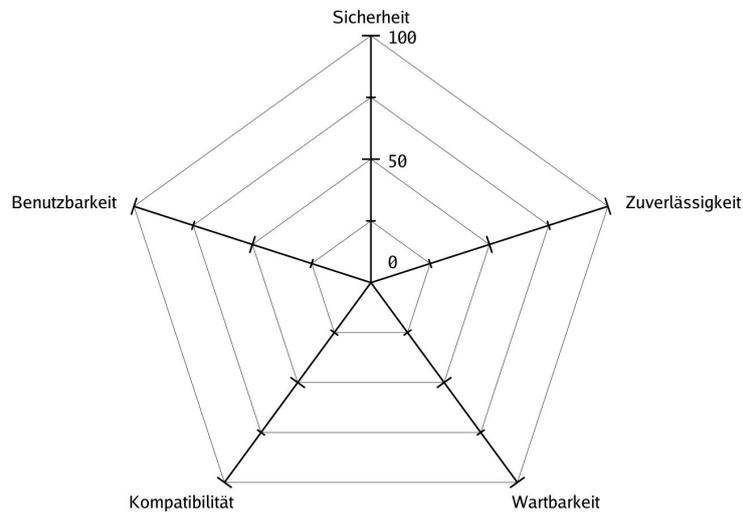
embarc.de

Softwarearchitektur in Eigenregie bewerten

43

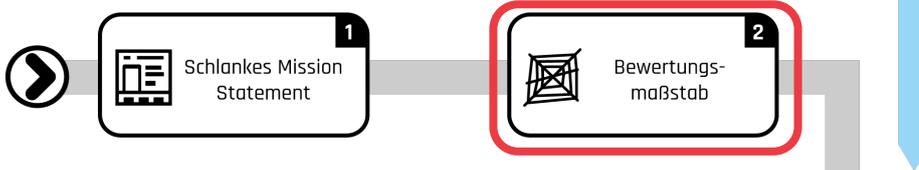


Ergebnis nach diesem Schritt (Beispiel)



Verstehe, was Dich speziell macht

Verstehe was Dich speziell macht



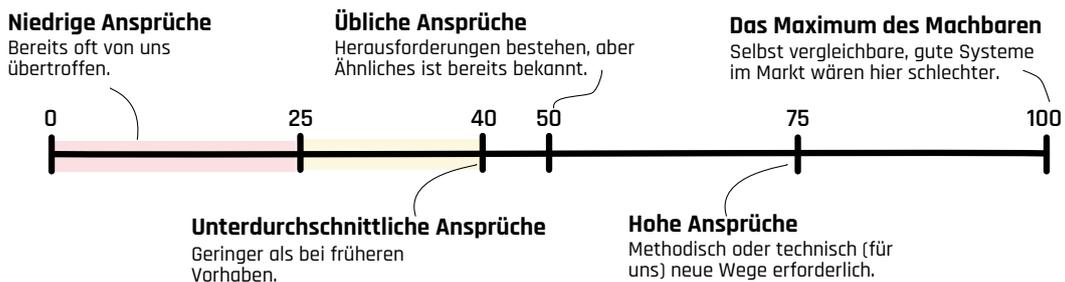
Was ist zu tun?

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



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.



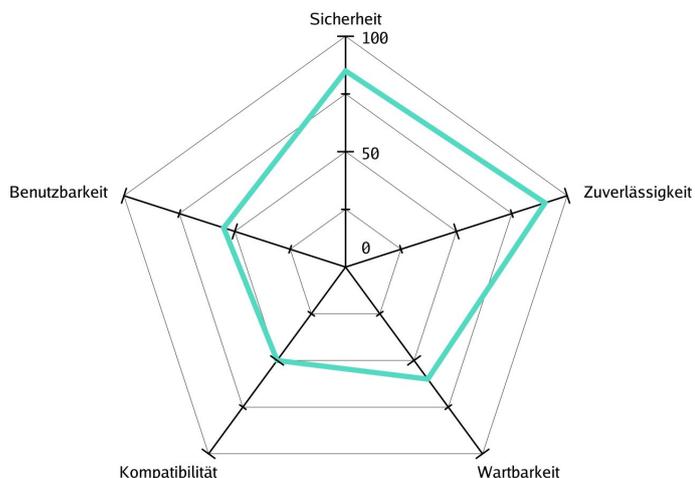
Hinweis: Es handelt sich hier um eine Einschätzung, nichts Messbares.

46

embarc.de

Softwarearchitektur in Eigenregie bewerten

Bewertungsmaßstab einzeichnen



LASR-Ergebnisdiagramm

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

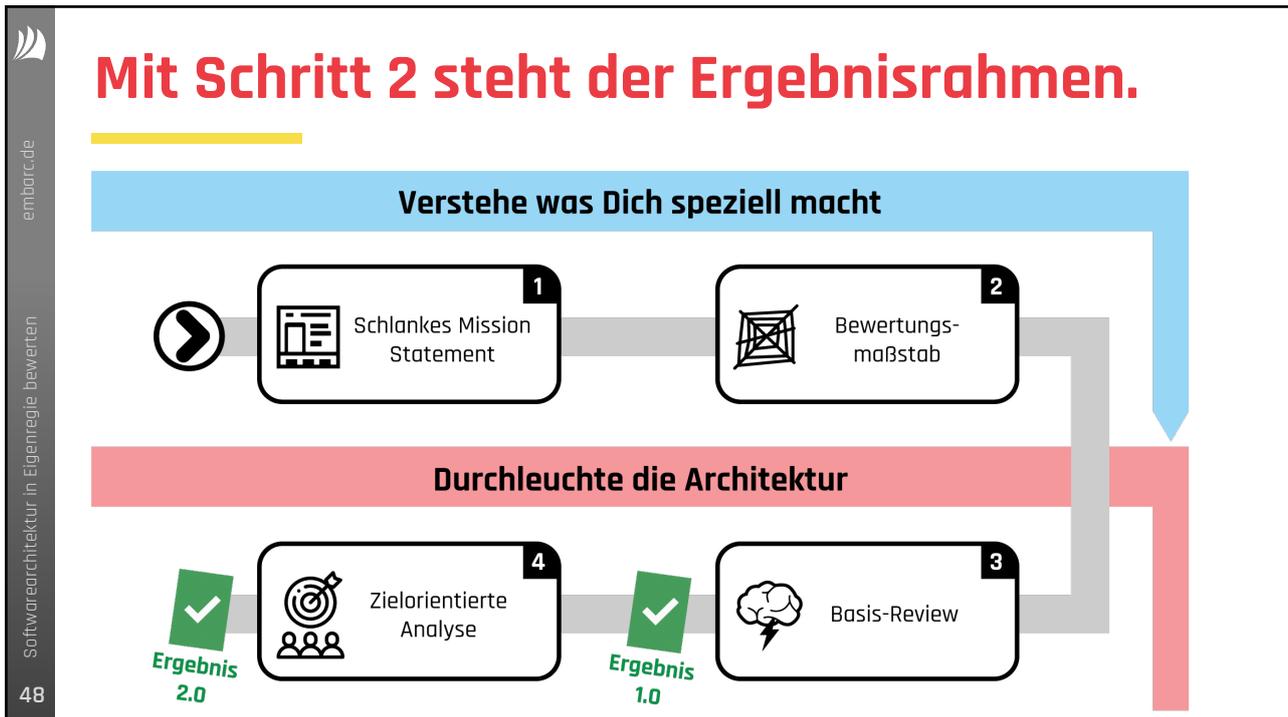
Die Zielwerte spannen darauf eine grüne Linie auf.

← Beispiel

47

embarc.de

Softwarearchitektur in Eigenregie bewerten



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



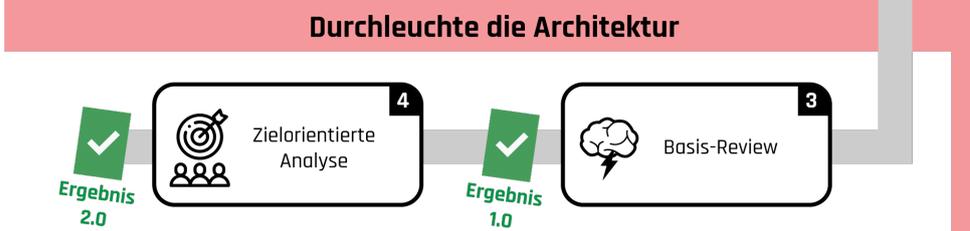


Durchleuchte die Architektur

embarc.de

Softwarearchitektur in Eigenregie bewerten

50



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

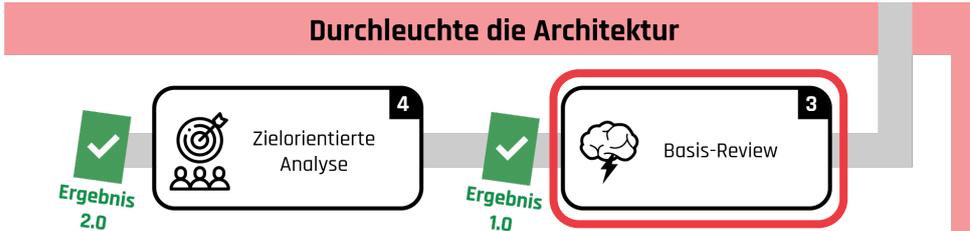


Durchleuchte die Architektur

embarc.de

Softwarearchitektur in Eigenregie bewerten

51



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



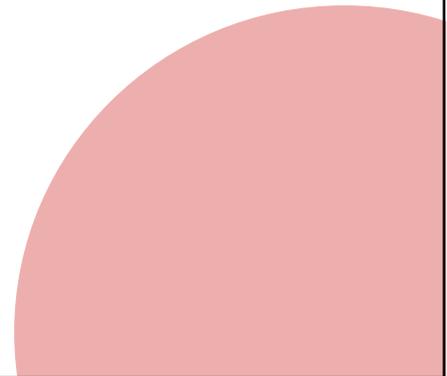
Produziert ein erstes Ergebnis in Form konkreter **Risiken**, die der Zielerreichung im Weg stehen.

Methodischer Ansatz

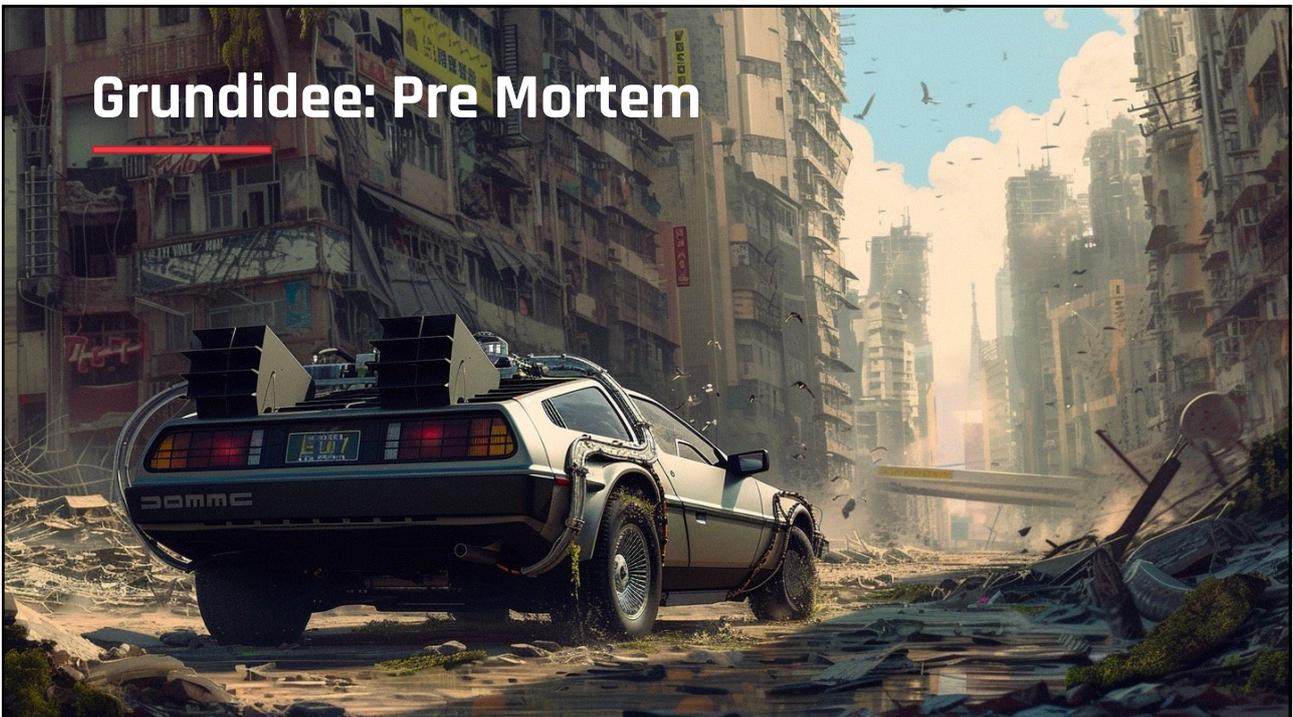
Brainstorming, angelehnt an **Pre-Mortem**

Ergebnisse des Schrittes

"Abweichung" vom Ziel, Ist-Einschätzung



Grundidee: Pre Mortem

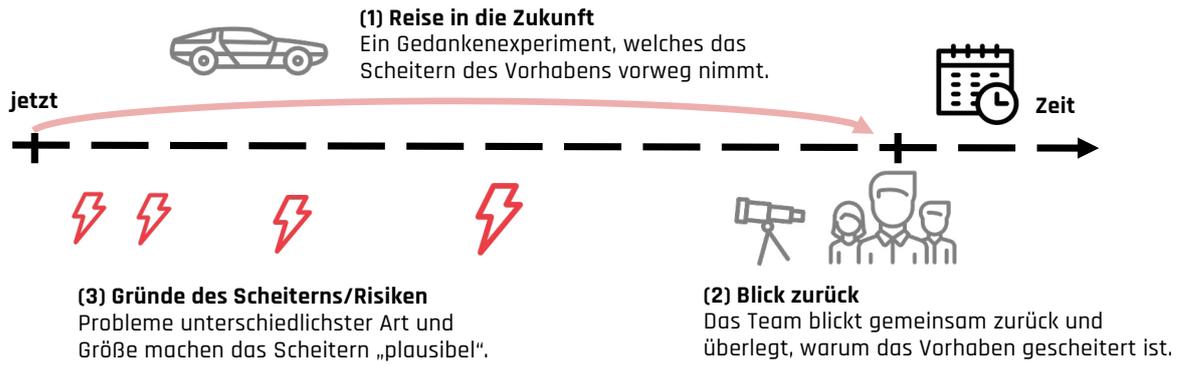


Was ist ein Pre-Mortem?

embarc.de

Softwarearchitektur in Eigenregie bewerten

54



Die 8 Risiko-Kategorien von LASR

<p>Softwarelösung 1</p> <ul style="list-style-type: none"> • Unpassende Technologien • Komplexe Lösungen • Unausgereifte Fremdlösungen • Behindernde Frameworks / Libraries • Strukturprobleme • ... 	<p>Kompetenz und Erfahrung 2</p> <ul style="list-style-type: none"> • Fehlendes oder isoliertes Wissen • Zu kleine Lernfenster • Fehlendes Prozessverständnis • Tool-Probleme • Schwere Schätzbarkeit • ... 	<p>Zielsetzungen und Erwartungen 3</p> <ul style="list-style-type: none"> • Unrealistische Ziele • Überzogene Erwartungen • Knappe Deadlines • Instabile Anforderungen • Fehlender Kundenkontakt • ... 	<p>Fremdsysteme und Plattformen 4</p> <ul style="list-style-type: none"> • Instabile Plattformen • Fehleranfällige Fremdsysteme • Unpassende Buy / Make Wahl • Lizenzschwierigkeiten • Vendor Lock-in • Integrationsprobleme • ...
<p>Altsysteme und Altlasten 5</p> <ul style="list-style-type: none"> • Legacy-Behinderungen • Innovationsstau • Zerbrochene Lösungen • Fehlende Tests • Wenig Dokumentation • Fehlendes Verständnis • ... 	<p>Organisation und Prozesse 6</p> <ul style="list-style-type: none"> • Geschwollene Prozesse • Behindernde Rollenmodelle • Organisationsgrenzen • Politik • Einengende Standards • Isolierte Entscheidungskompetenzen • ... 	<p>Betrieb und Deployment 7</p> <ul style="list-style-type: none"> • Blockierende Prozesse • Geringe Automatisierung • Wenig Feedback • Fehlende Betriebs- / Ausfallkonzepte • Tool-Probleme • ... 	<p>Weiche Faktoren 8</p> <ul style="list-style-type: none"> • Unregelmäßigkeiten • Konflikte • Fehlende Disziplin • Kommunikationsbarrieren • Rollenmatrizen • Unpassende Kultur • ...

Typische Risikotheemen als Ideengeber

<p>Zu hohe Komplexität der Lösung</p> <p>Hat die Domäne eine sehr hohe Komplexität? Gibt es unüberlegte, schnelle Lösungen oder fehlende Abstraktionen? Komplexität gefährdet den Überblick bzw. Wartbarkeit, Korrektheit, Sicherheit ...</p> <p>Softwarelösung 1.1</p>	<p>Unpassende Lösungs-Strukturierung</p> <p>Folgt die Softwarestruktur der Domäne? Sind andere technische oder organisatorische Einflüsse (Canway...) gut aufgegriffen? Falls nicht könnte Wartbarkeit, Zuverlässigkeit etc. leiden.</p> <p>Softwarelösung 1.2</p>	<p>Inadäquates Daten-Handling</p> <p>Ist die Abloge, 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?</p> <p>Softwarelösung 1.3</p>	<p>Problematische Konzepte & Technologien</p> <p>Passen die eingesetzten Muster und Konzepte zu den Zielen? Sind sie konsistent angewendet? Sind die verwendeten Technologien und Frameworks passend und etabliert?</p> <p>Softwarelösung 1.4</p>
--	---	--	--

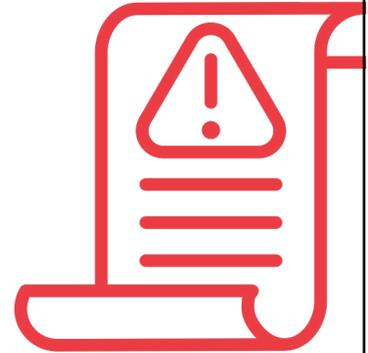


Brainstorming-Unterstützung: Das LASR-Kartenset hält insgesamt 32 konkrete Risikokarten als Ideengeber parat, 4 in jeder der 8 Kategorien.



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)



embarc.de

Softwarearchitektur in Eigenregie bewerten

60



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.“



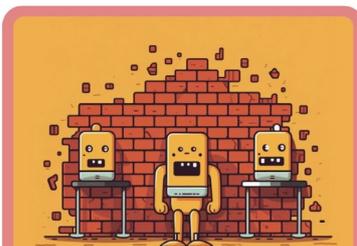
Hypothetisches Risiko (2)



embarc.de

Softwarearchitektur in Eigenregie bewerten

61



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?

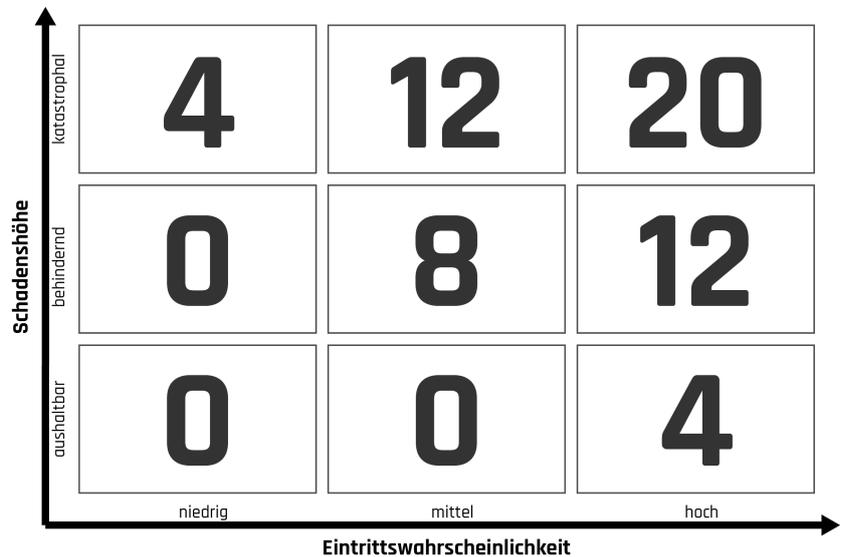
Organisation & Prozesse 6.4

Private Kommunikation scannen

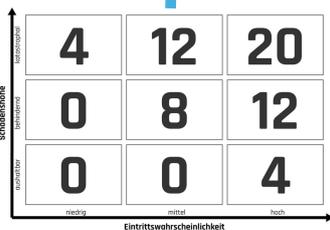
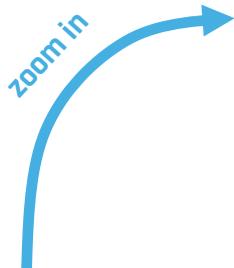
„Die europäische Gesetzgebung hat Messenger-Anbieter verpflichtet, sämtliche private Kommunikation und Dateien zu durchleuchten. Threema muss daraufhin die Lösung für private Nutzer komplett neu denken.“



Risiken einordnen



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:



Risikopunktzahl: (Wert pro Risiko)

12

Qualitätsziel 1:

Risikopunktzahl: (Wert pro Risiko)

12

Qualitätsziel 2:

Risikopunktzahl: (Wert pro Risiko)

12

Qualitätsziel 3:

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

Risikopunktzahl: (pro Risiko für jedes betroffene Qualitätsziel)

12

Risikopunktzahl: (Wert pro Risiko)

12

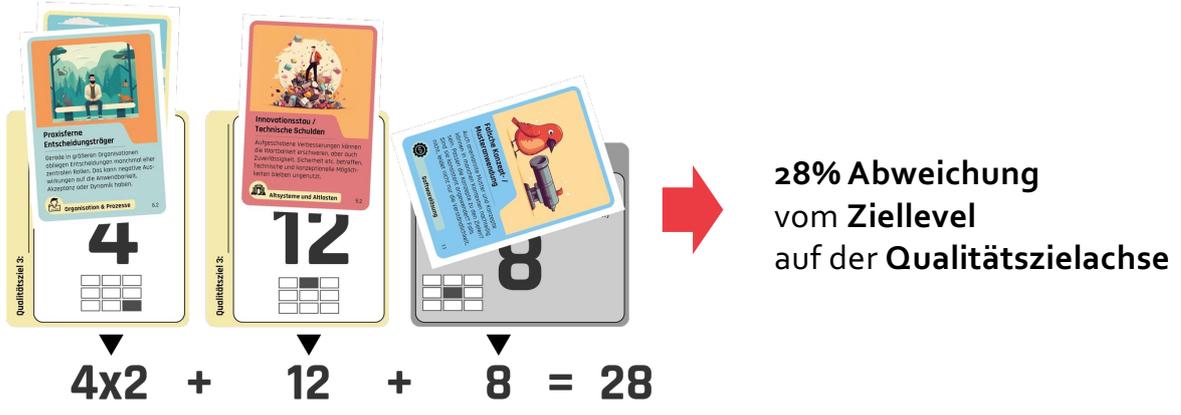
Qualitätsziel 4:

Risikopunktzahl: (Wert pro Risiko)

12

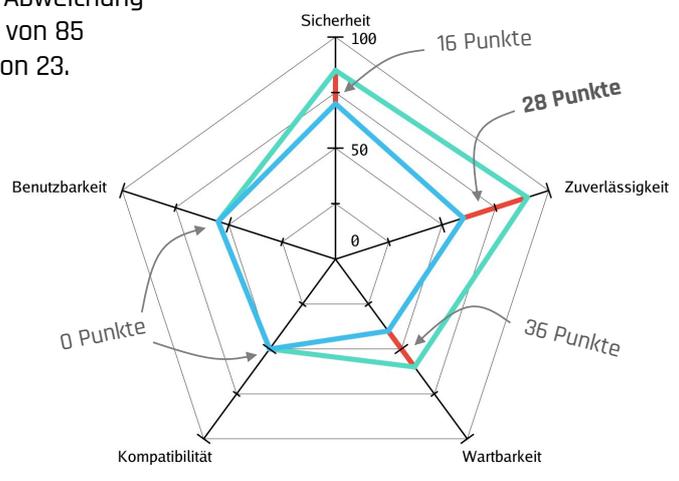
Qualitätsziel 5:

Von Risiken zu „Lücken“



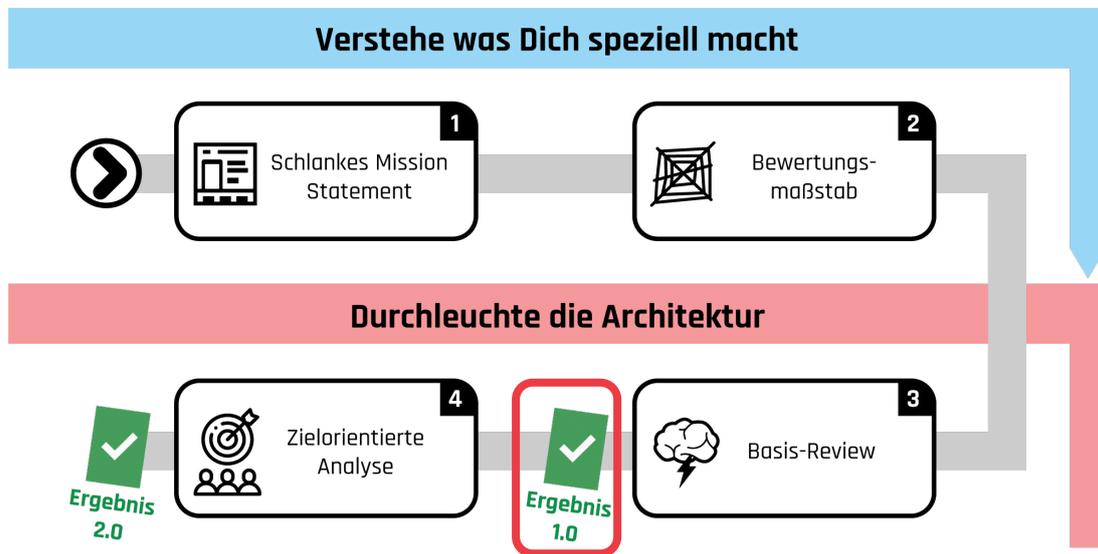
Abweichungen einzeichnen

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

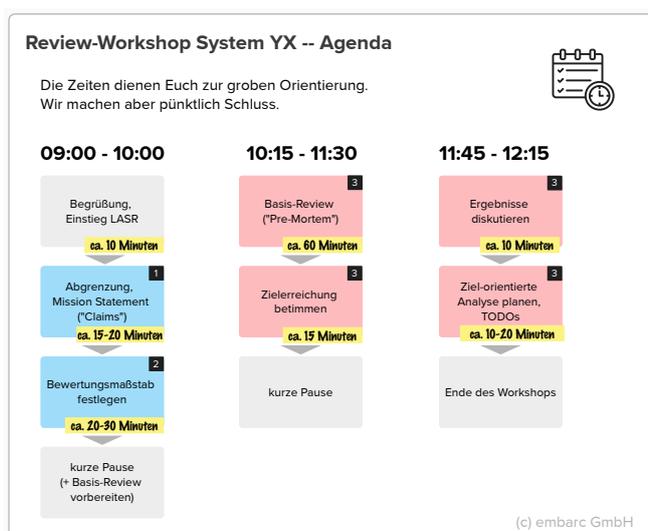


embarc.de

Softwarearchitektur in Eigenregie bewerten

70

LASR an einem Vormittag



← Beispiel-Agenda

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

embarc.de

Softwarearchitektur in Eigenregie bewerten

71

04.

Erhöhe die Konfidenz (bei Bedarf)

01. Warum leichtgewichtig bewerten?

02. Verstehe, was Dich speziell macht

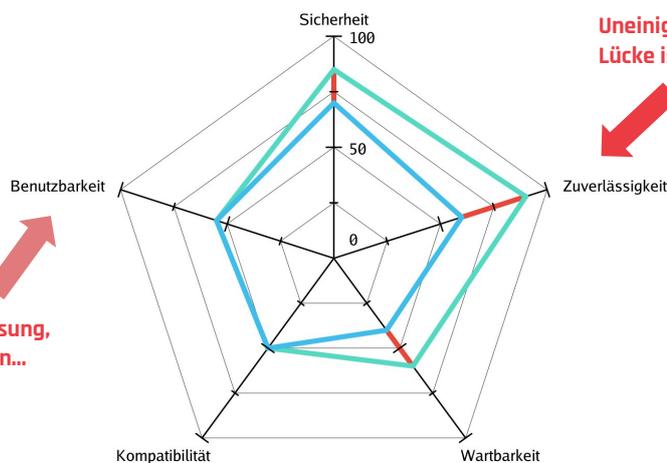
03. Durchleuchte die Architektur

04. Erhöhe die Konfidenz (bei Bedarf)

05. Weitere Informationen



Fokus: Zielorientierte Analyse (Beispiele)

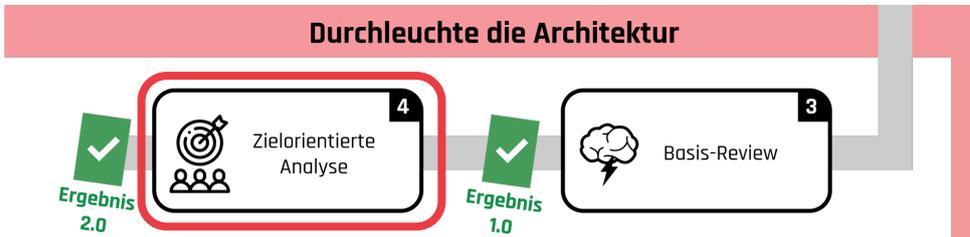


Keine Stärken in der Lösung,
Kein explizites Vorgehen...
Trotzdem keine Lücke?

Uneinigkeit!
Lücke ist strittig



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 →

Zielchse

Lücke (Schritt 3):

36

Lücke geschäft:

20

Konkrete Risiken

Welche Risiken sind der Zielchse zugeordnet?

- 1. Nebenstellen wären bei Notfällen abgeköpft, möglicherweise Problem? keine QA betroffen, keine Mitigation
- 2. Verändergeschichte bei Problem, Aussagen nicht überprüfbar? keine QA betroffen, keine Mitigation
- 3. QA zu Mitwirkungsveränderung angewiesen? keine QA betroffen, keine Mitigation
- 4. Erkennung von Audit und Linische ist manuell? keine QA betroffen, keine Mitigation

1C	2C	3C
1P	2P	3P
1A	2A	3A

1C Risiko-Einschätzung

Qualitätsaussagen

Welche Aussagen illustrieren die Zielchse?

- 1. Kernleistungen: Messschritt auf 80-90% → Kosten/Produktion
- 2. Die Verbindung zum Servicecenter ist unterbrochen, die Anlage muss manuell für den Betrieb sein
- 3. Es handelt sich um einen kleinen, abgrenzbaren Teil des mit 100% - 95% - 90% arbeitenden Systems

Zentrale QAs - Qualitätsaussagen Technisches Risiko dass QA erreicht wird: High / Medium / Low

Stärken der Lösung

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

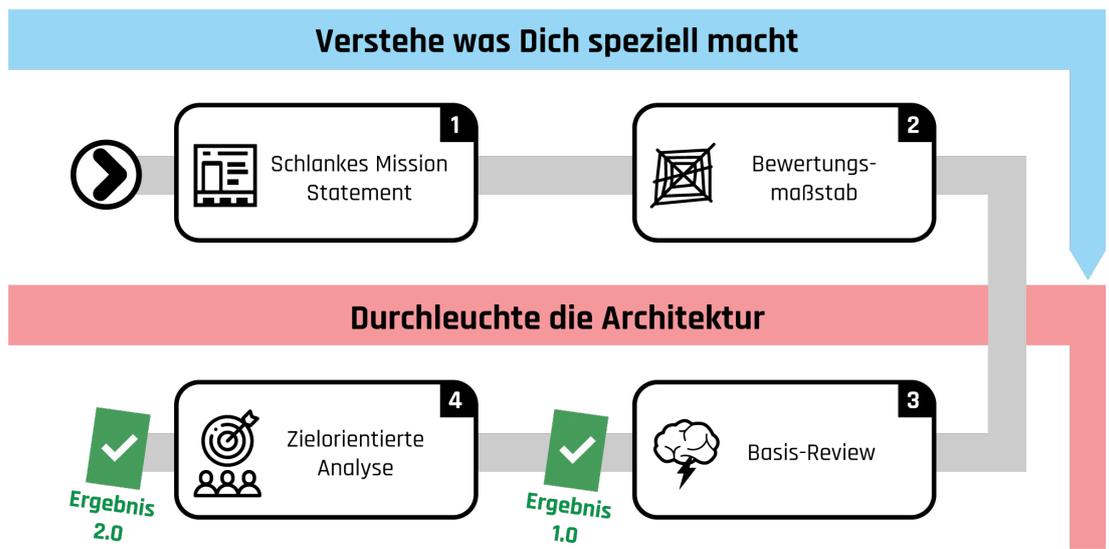
- Messungsbasierte Kommunikation (begrenzt, netz-fähig)
- Produktion ohne manuelle Verbindung möglich (Überschneidung bei 80% und 90%)
- Verbindung der VCS auch abgeköpft + haben gehen muss verhindern
- Verbindungs-komponente kann 3000 ohne Einsatz überbrücken

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 VCS

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.

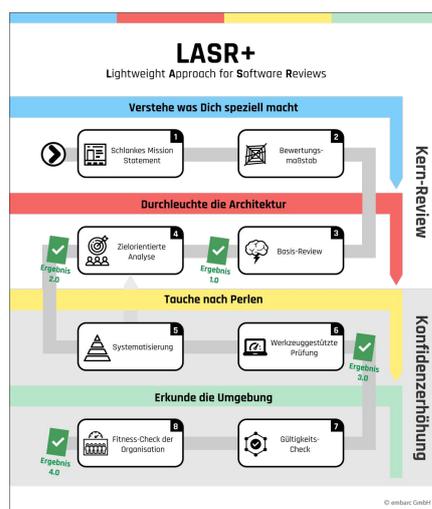
Wo fangen wir an?

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.

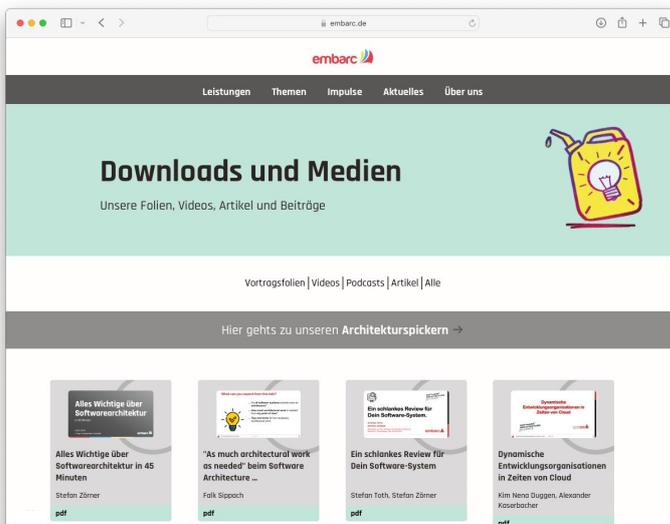
05.

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**



Folien als PDF zum Download



➔ embarc.de/download/

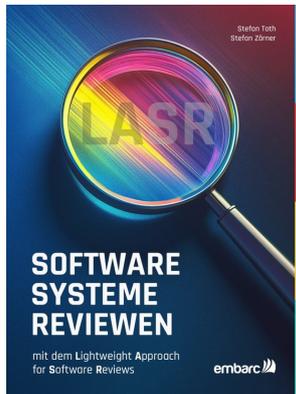


Buchtipp zum Thema

embarc.de

Softwarearchitektur in Eigenregie bewerten

82



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/

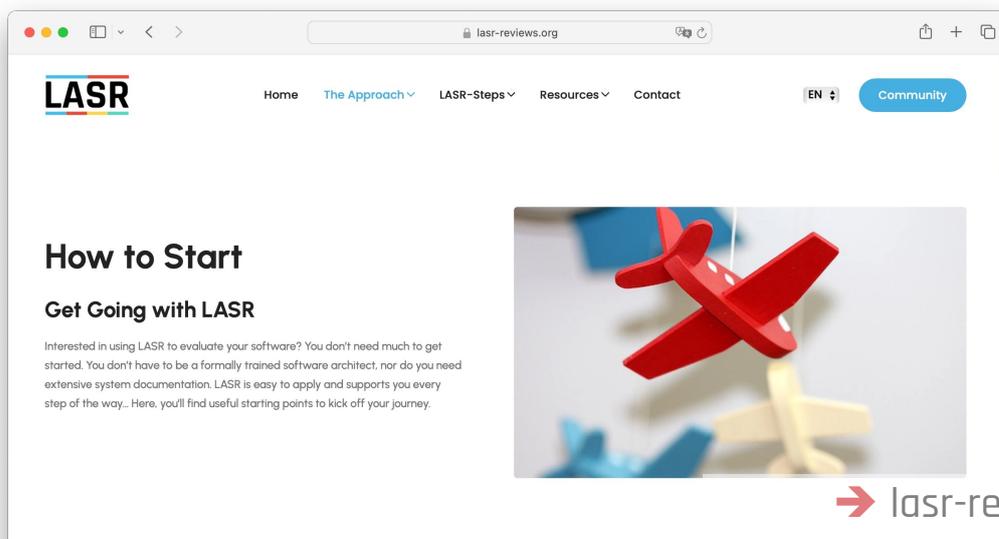


„Offizielle“ LASR-Webseite (EN + DE)

embarc.de

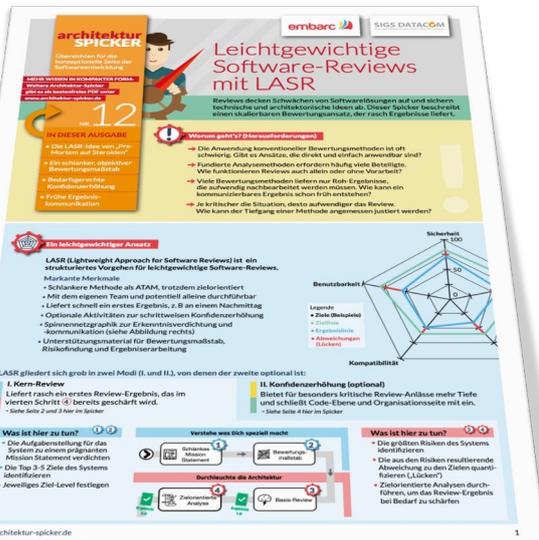
Softwarearchitektur in Eigenregie bewerten

83



→ lasr-reviews.org

Architektur-Spicker zum Thema ...



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

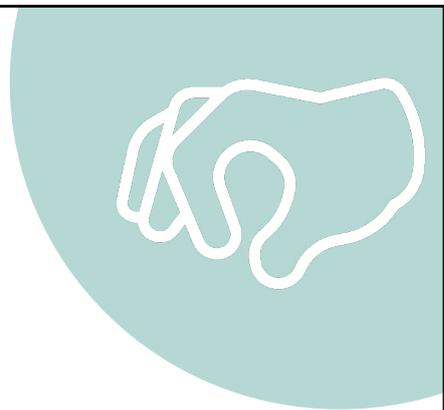
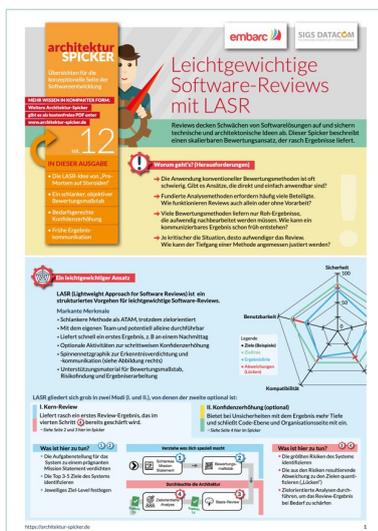
Architektur-Spicker #12 Leichtgewichtige Software Reviews mit LASR



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

am 08. Dezember 2025 | remote

Kim Duggen Oliver Zeigermann Alexander Kaserbacher Felix Kammerlander Cosima Laube

Arnold Franke Falk Sippach Stefan Zörner Stefan Toth

**Workshops, neue Perspektiven
& Plaudern in lockerer Punsch-Atmosphäre**

**Infos, Programm
& Anmeldung:**

Vielen Dank.

Ich freue mich auf Eure Fragen!

Stefan.Zoerner@embarc.de

[linkedin.com/in/stefan-zoerner](https://www.linkedin.com/in/stefan-zoerner)

[@StefanZoerner@mastodon.social](https://mastodon.social/@StefanZoerner)

embarc.de

Softwarearchitektur in Eigenregie bewerten

87