Das Wichtigste zu Architektur-Reviews in 30 Minuten

Stefan Zörner, embarc

Fokus-Event Softwarearchitektur bei Conciso
Dortmund, 9. Oktober 2024

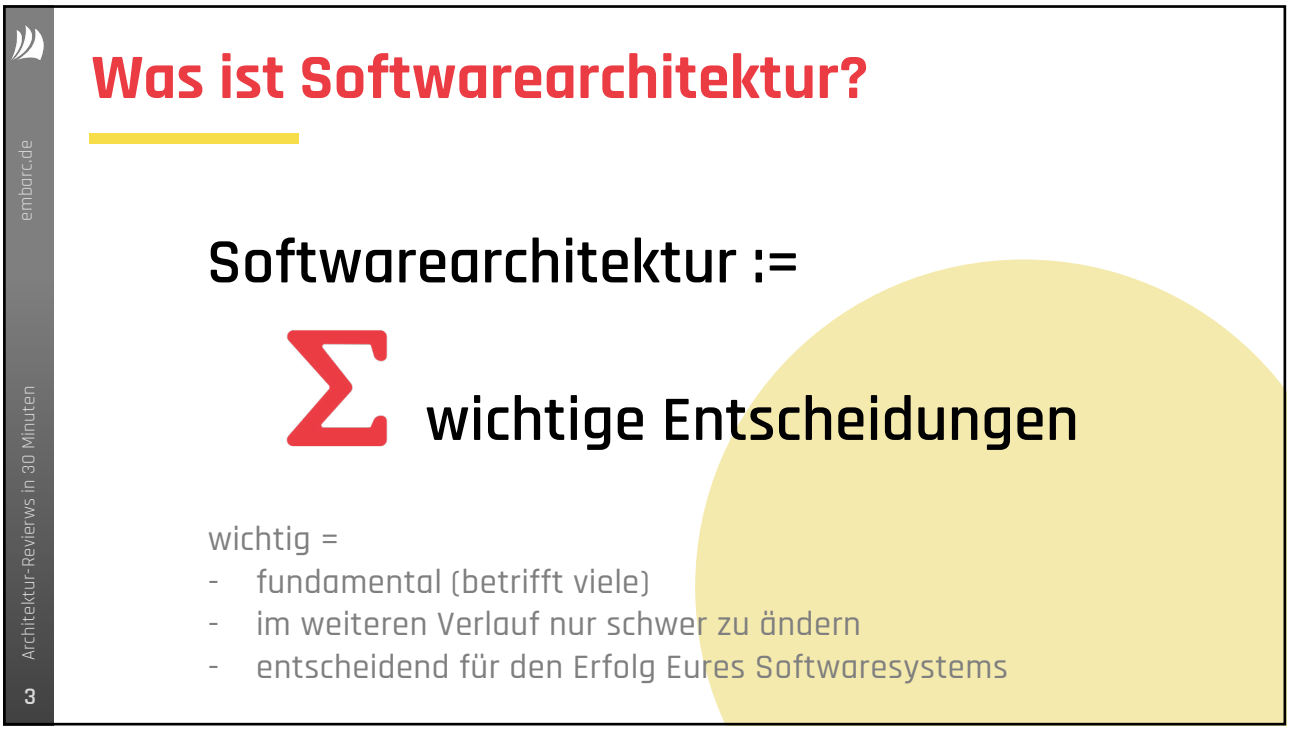
Stefan Zörner

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

Schwerpunkte

- Softwarearchitektur (Entwurf, Bewertung, Dokumentation)
- Cloud-Technologien

Was ist ein Architektur-Review und wann ist es nützlich?



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, ...

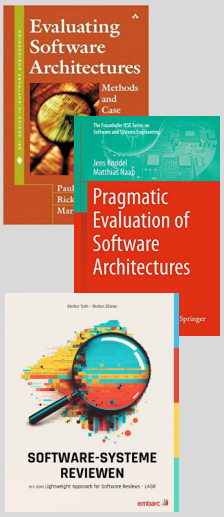
Was ist ein Review?

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.

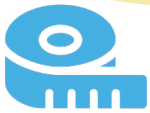


Kernelemente eines Reviews

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?

embarc.de
Architektur-Reviews in 30 Minuten

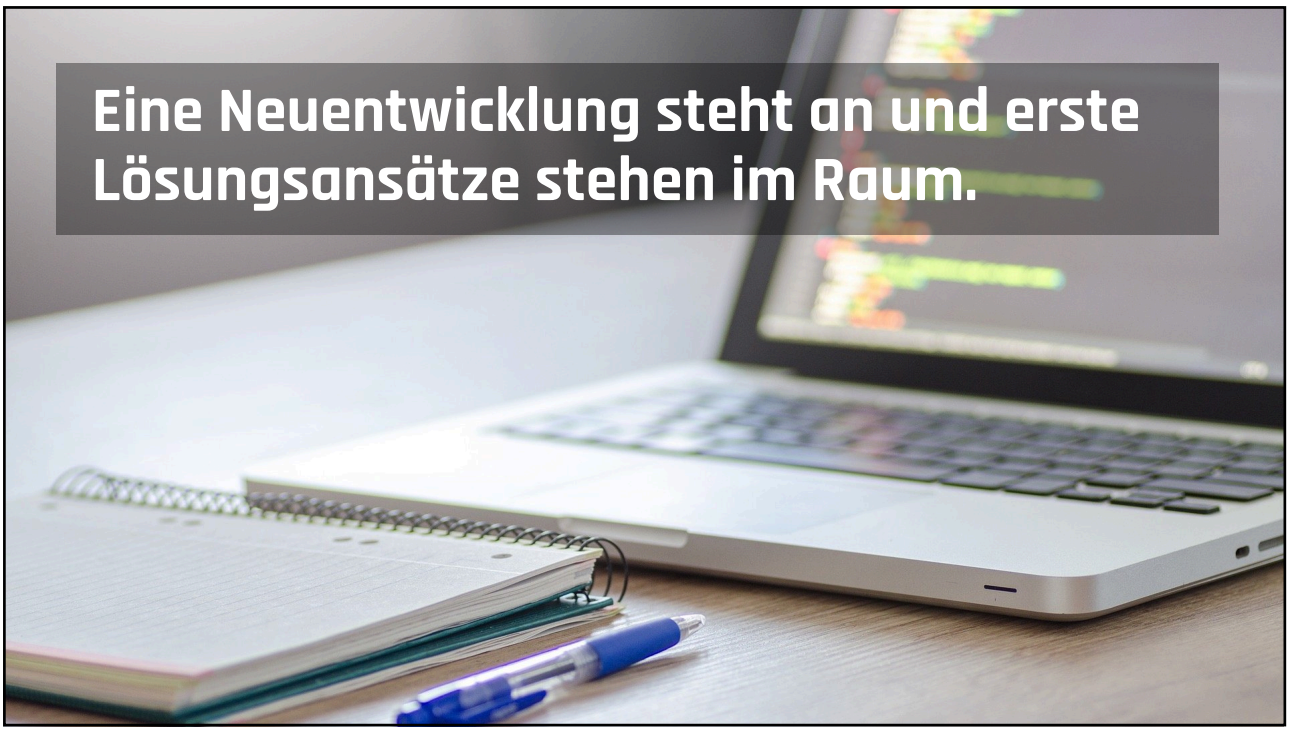
Kernelemente eines Reviews

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**?

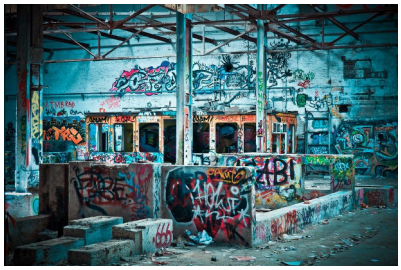
Typische Anlässe (... findet Ihr Euch wieder?)

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?)

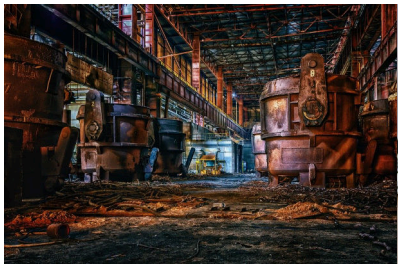


Anlass 3 („Unruhe“)

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.

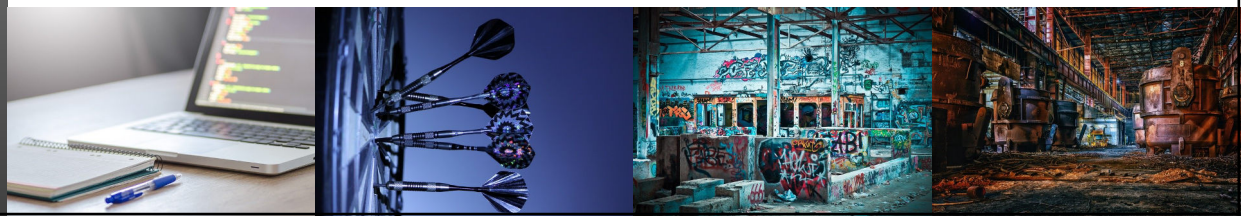
Leitfrage:
Wie wählt Ihr passende Lösungsansätze **nachvollziehbar** aus?

Das Leistungsversprechen von Reviews

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

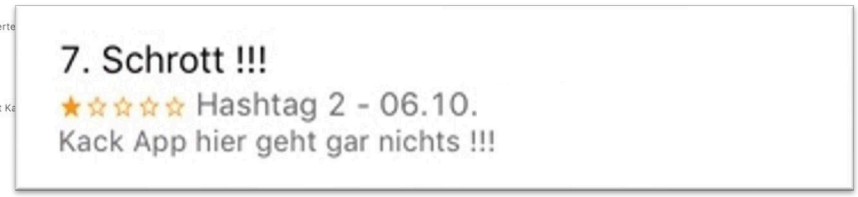
Konkret: Software-Reviews …

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



Wie funktioniert klassische Architekturbewertung?

Ad-hoc-Review



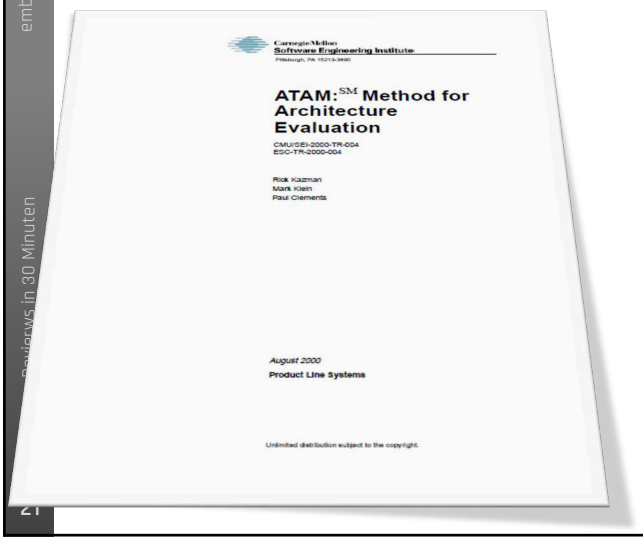
Bewertungsmethoden (Auswahl)

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

Grober Ablauf von ATAM

Phase 0: Vorbereitung

Phase 1: Architektur-zentrierte Analyse

Präsentieren

- Schritt 1: ATAM präsentieren
- Schritt 2: Geschäftsziele präsentieren
- Schritt 3: Architektur präsentieren

Untersuchen

- Schritt 4: Architekturansätze identifizieren
- Schritt 5: Utility Tree erstellen
- Schritt 6: Architekturansätze analysieren

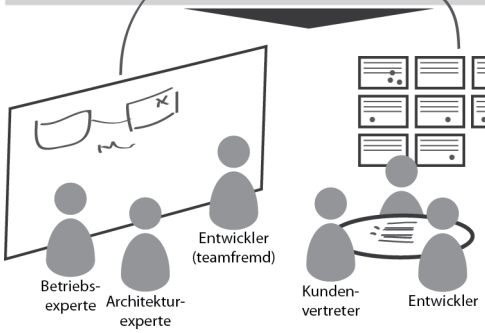
Phase 2: Stakeholder-zentrierte Analyse

Phase 3: Nachbereitung

Architekturskizzen
Messergebnisse
Annahmen
Entscheidungen

Vision & Ziele
Rahmenbedingungen
Qualitätsmerkmale
Szenarien

Inputs



Workshop

Transparenz
Feedback
Klarheit
...

Kompromisse
Risiken
Nicht-Risiken
ToDos

Outputs

Quelle der Abbildung: Stefan Toth, "Vorgehensmuster für Softwarearchitektur", Hanser

Herausforderungen …

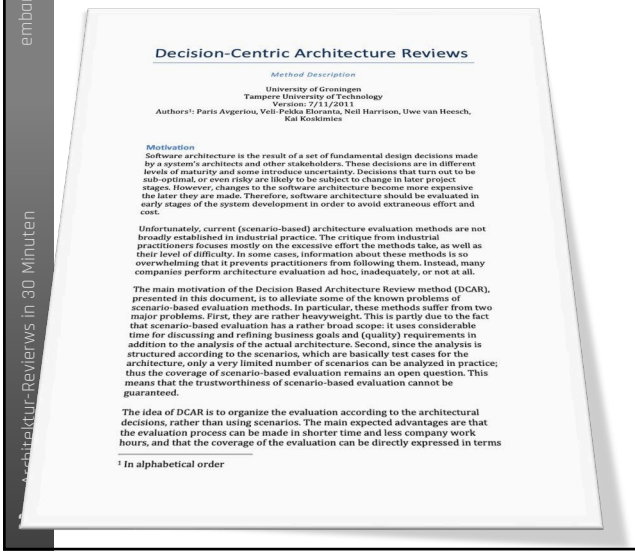
Die Anwendung fundierter Bewertungsmethoden ist mitunter schwierig.

- 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**.

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

DCAR



Decision-Centric Architecture Reviews

Method Description

University of Groningen
Tampere University of Technology
Version: 7/11/2011
Authors[1]: Paris Avgeriou, Veli-Pekka Eloranta, Neil Harrison, Uwe van Heesch, Kai Koskimies

Motivation

Software architecture is the result of a set of fundamental design decisions made by a system's architects and other stakeholders. These decisions are in different levels of maturity and some introduce uncertainty. Decisions that turn out to be sub-optimal, or even risky are likely to be subject to change in later project stages. However, changes to the software architecture become more expensive the later they are made. Therefore, software architecture should be evaluated in early stages of the system development in order to avoid extraneous effort and cost.

Unfortunately, current (scenario-based) architecture evaluation methods are not broadly established in industrial practice. The critique from industrial practitioners focuses mostly on the excessive effort the methods take, as well as their level of difficulty. In some cases, information about these methods is so overwhelming that it prevents practitioners from following them. Instead, many companies perform architecture evaluation ad hoc, inadequately, or not at all.

The main motivation of the Decision Based Architecture Review method (DCAR), presented in this document, is to alleviate some of the known problems of scenario-based evaluation methods. In particular, these methods suffer from two major problems. First, they are rather heavyweight. This is partly due to the fact that scenario-based evaluation has a rather broad scope: it uses considerable time for discussing and refining business goals and (quality) requirements in addition to the analysis of the actual architecture. Second, since the analysis is structured according to the scenarios, which are basically test cases for the architecture, only a very limited number of scenarios can be analyzed in practice; thus the coverage of scenario-based evaluation remains an open question. This means that the trustworthiness of scenario-based evaluation cannot be guaranteed.

The idea of DCAR is to organize the evaluation according to the architectural decisions, rather than using scenarios. The main expected advantages are that the evaluation process can be made in shorter time and less company work hours, and that the coverage of the evaluation can be directly expressed in terms

[1] In alphabetical order

Decision-Centric Architecture Review

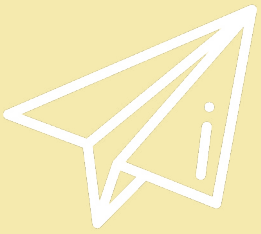
- Ursprung: Universitäten von Groningen (NL), Tampere (FI), 2011
- Initial ähnlicher Ablauf zu ATAM, produziert auch ähnliche Ergebnisse
- Kommt mit weniger Vorbereitung aus.
- Geht in der Analyse von den **(Architektur-)Entscheidungen** aus

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

Wie sähe ein leichtgewichtiges Review eurer Softwarelösung aus?

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.

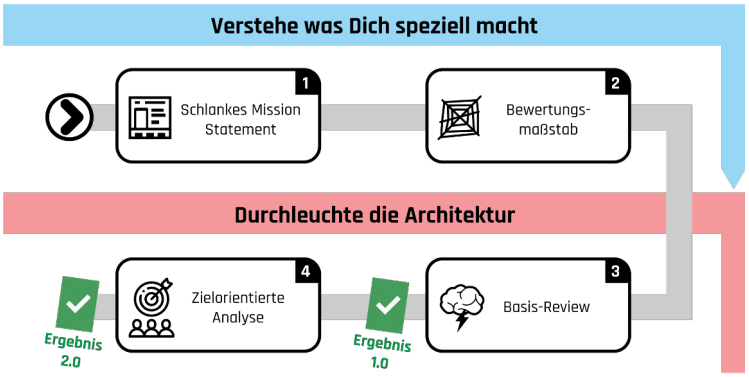
- Entwicklungsteam führt Review **allein** und **ohne Vorbereitung** durch.
- Bereits nach einem halben Tag liegt ein **kommunizierbares Ergebnis** vor.

Ein schlanker Ansatz

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

LASR Kern-Review

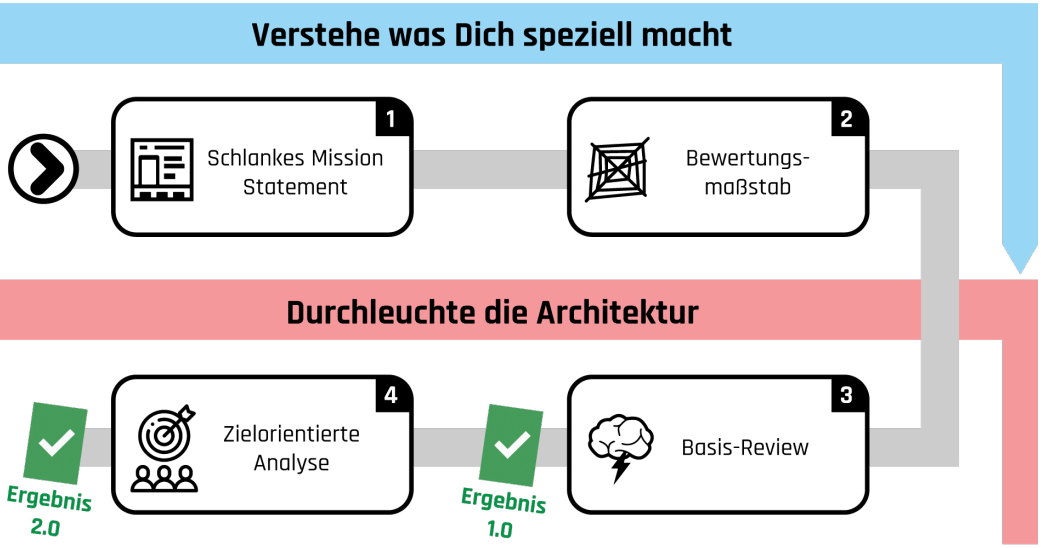
- 2 Tätigkeiten
- 4 Schritte
- Ein frühes erstes Ergebnis



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 Bewertungs-maßstab, Risikofindung und Ergebniserarbeitung

Tätigkeiten und Schritte im Kern-Review



LASR an einem Vormittag

Review-Workshop System YX -- Agenda

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

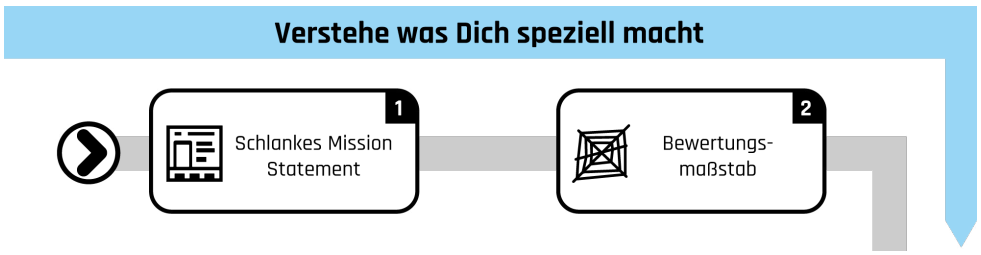
09:00 - 10:00	10:15 - 11:30	11:45 - 12:15
Begrüßung, Einstieg LASR — ca. 10 Minuten	3 Basis-Review ("Pre-Mortem") — ca. 60 Minuten	3 Ergebnisse diskutieren — ca. 10 Minuten
1 Abgrenzung, Mission Statement ("Claims") — ca. 15-20 Minuten	3 Zielerreichung betimmen — ca. 15 Minuten	3 Ziel-orientierte Analyse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

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

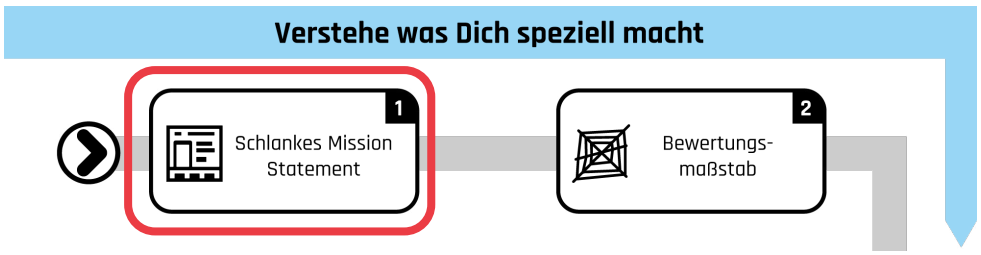
Für eine Bewertung oder ein Review braucht es mindestens

Einen Anlass
Warum bewerten wir?

Einen Gegenstand
Was bewerten wir?

Einen Maßstab
Wonach (wo gegen) bewerten wir?

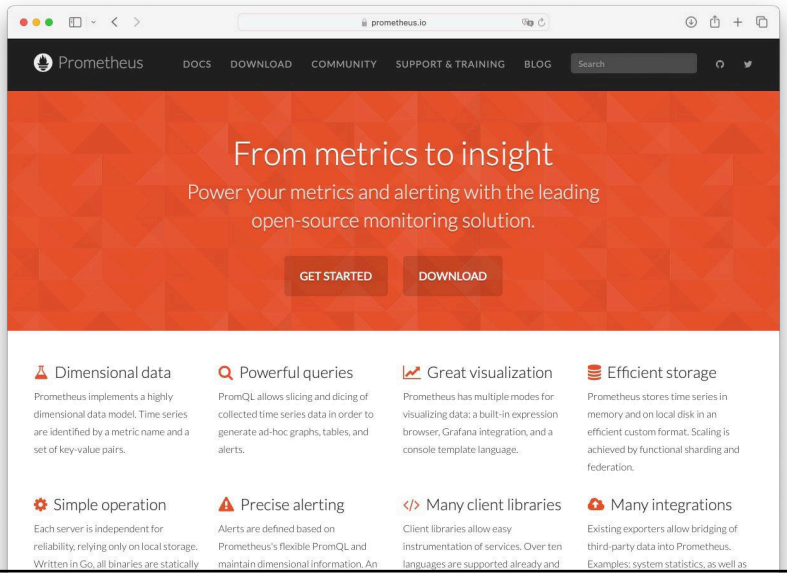
Verstehe, was Dich speziell macht



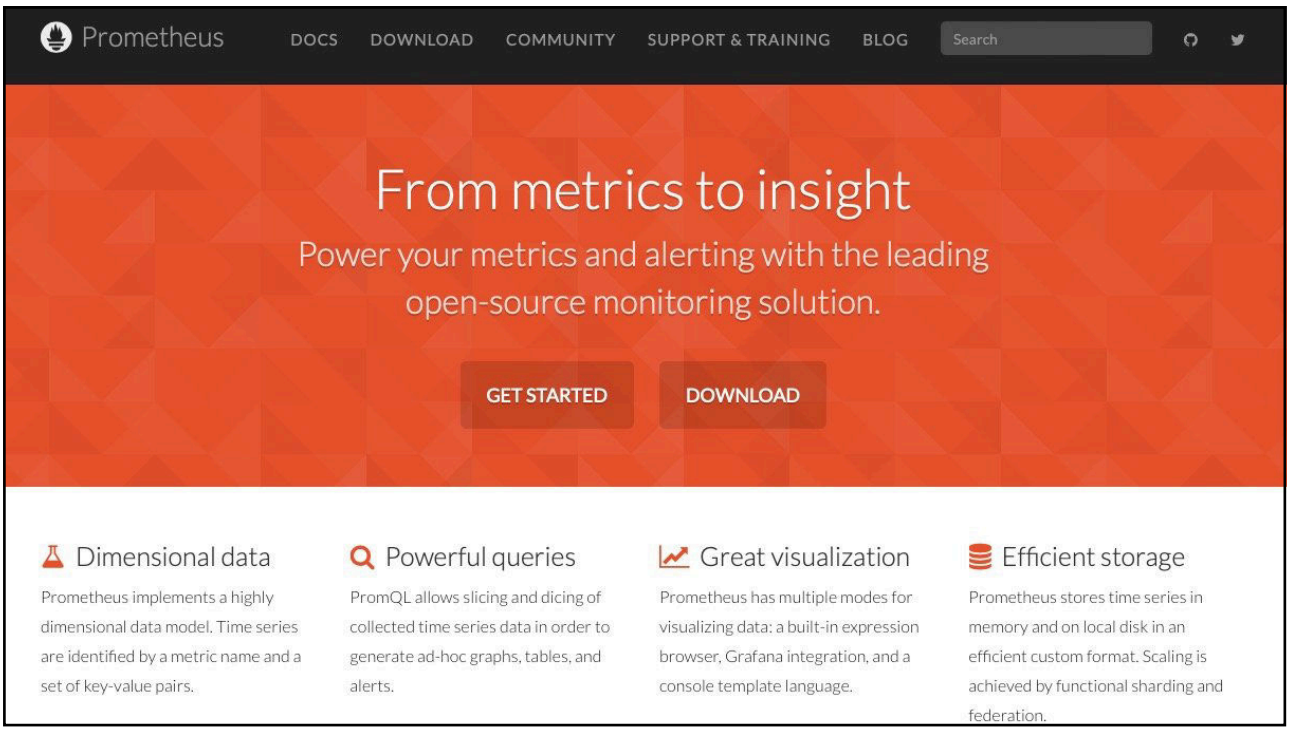
Was ist zu tun?

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

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.

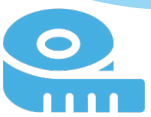

embarc.de
Architektur-Reviews in 30 Minuten
39
Erarbeiten im Workshop
LASR Schritt 1: Schlankes Mission Statement
Was würde prominent auf der Landing Page der zu betrachtenden Softwarelösung stehen?
1 Brainstorming
Leitfragen:
Was sind zentrale Features oder Eigenschaften unserer Software?
Was macht 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

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?

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? ...

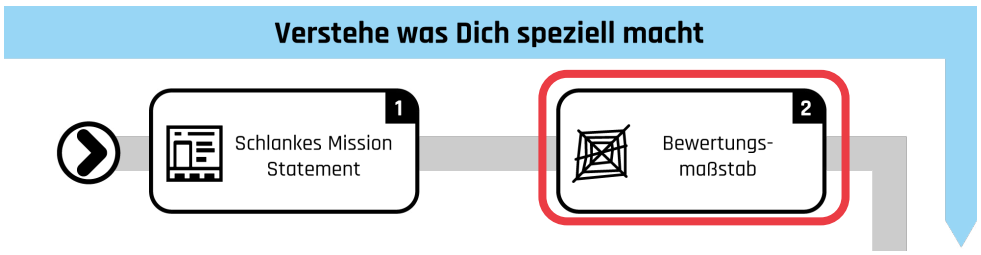
Sicherheit
(Security)
Ist das System sicher vor Angriffen? Sind Daten und Funktion vor unberechtigtem Zugriff geschützt? ...

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? ...

Verstehe, was Dich speziell macht



Was ist zu tun?

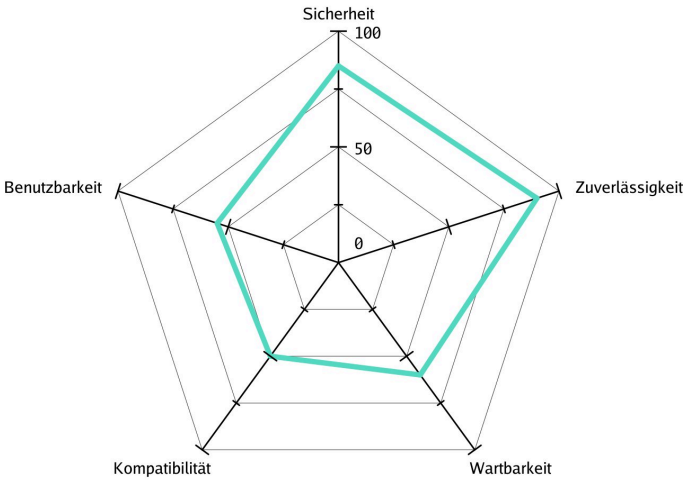
- Die Aufgabenstellung für das System zu einem prägnanten Mission Statement verdichten
- **Die Top 3-5 Ziele des Systems identifizieren**
- Jeweiliges Ziel-Level festlegen

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

Bewertungsmaßstab einzeichnen



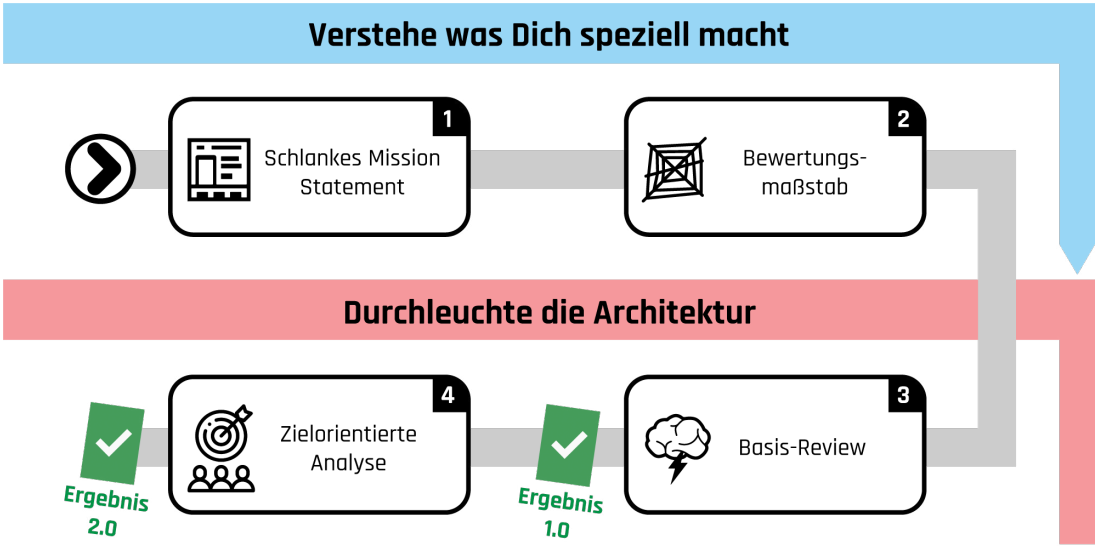
LASR-Ergebnisdiagramm

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

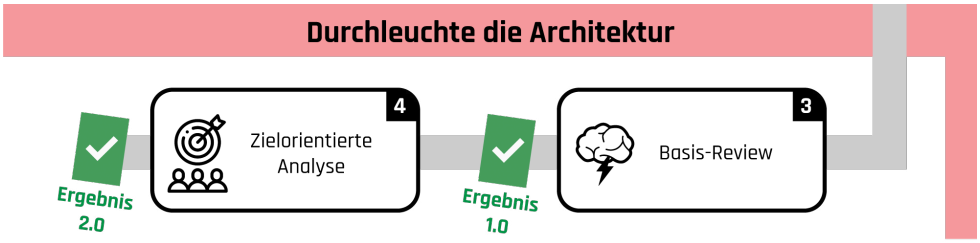
Die Zielwerte spannen darauf eine grüne Linie auf.

Beispiel

Mit Schritt 2 steht der Ergebnisrahmen.



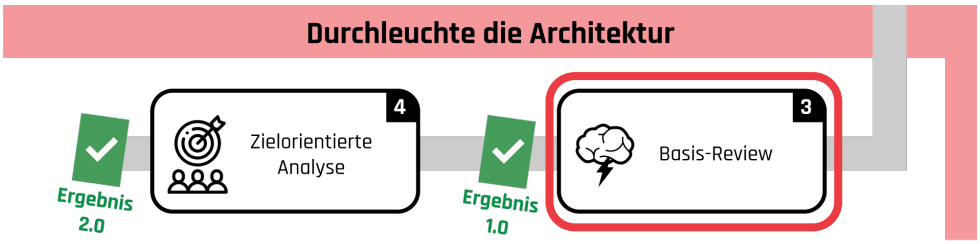
Durchleuchte die Architektur



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

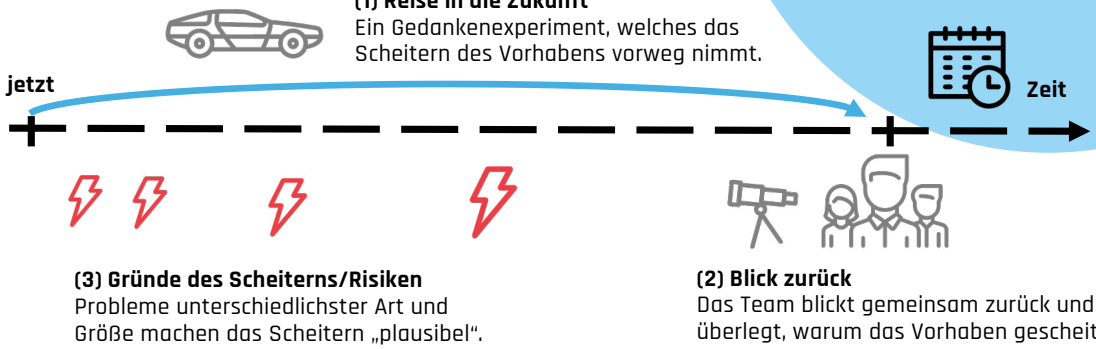
Durchleuchte die Architektur



Was ist zu tun?

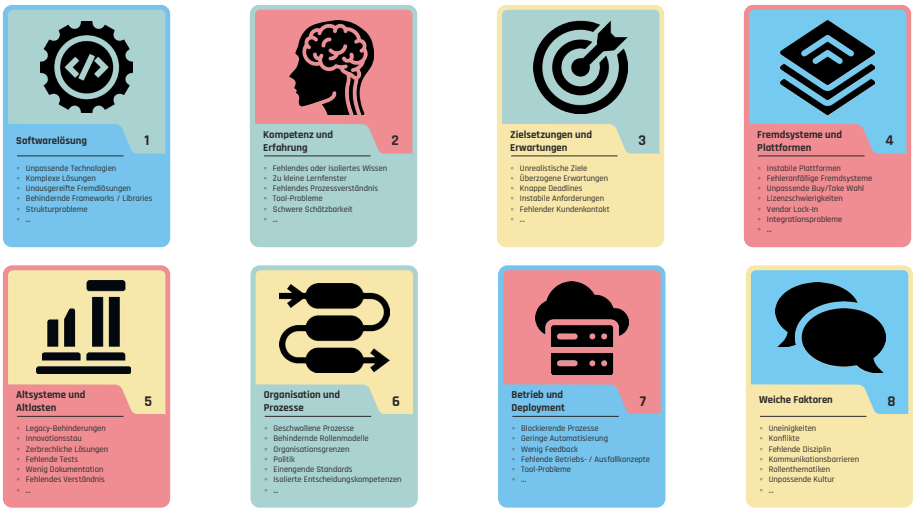
- **Die größten Risiken des Systems identifizieren**
- Die aus den Risiken resultierende Abweichung zu den Zielen quantifizieren („Lücken“)
- Zielorientierte Analysen durchführen, um das Review-Ergebnis bei Bedarf zu schärfen

Was ist ein Pre-Mortem?



- Pre-Mortems werfen einen kritischen Blick auf Vorhaben. Und zwar aus einer hypothetischen Zukunft, in der das Vorhaben gescheitert ist.
- Aus dieser Perspektive ergeben sich „Gründe des Scheiterns", die aus heutiger Sicht mögliche Risiken darstellen.

Die 8 Risiko-Kategorien von LASR

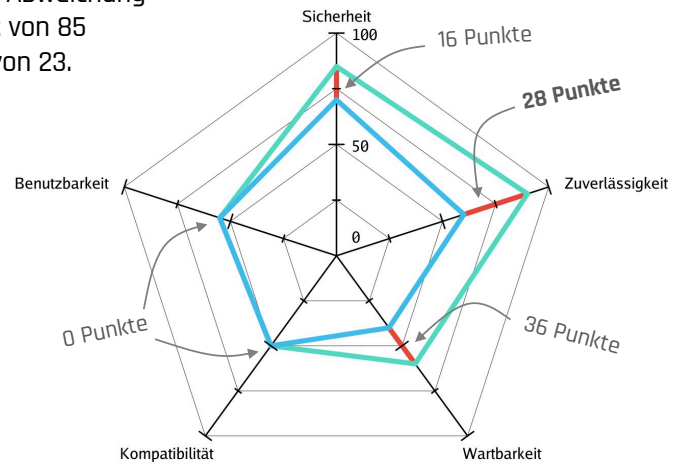


Typische Risikothemen als Ideengeber

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

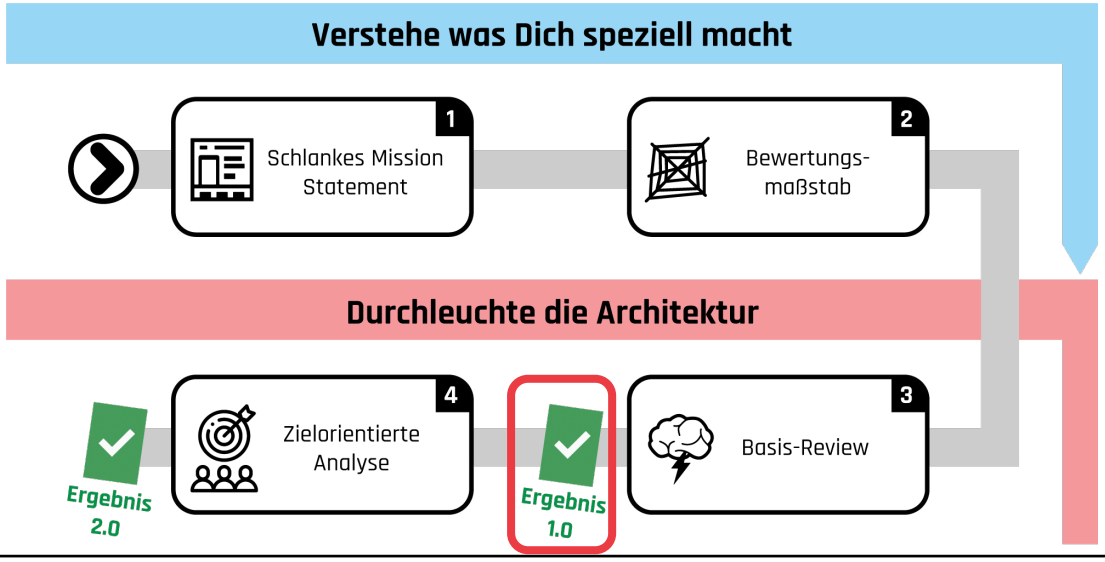
Abweichungen einzeichnen

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

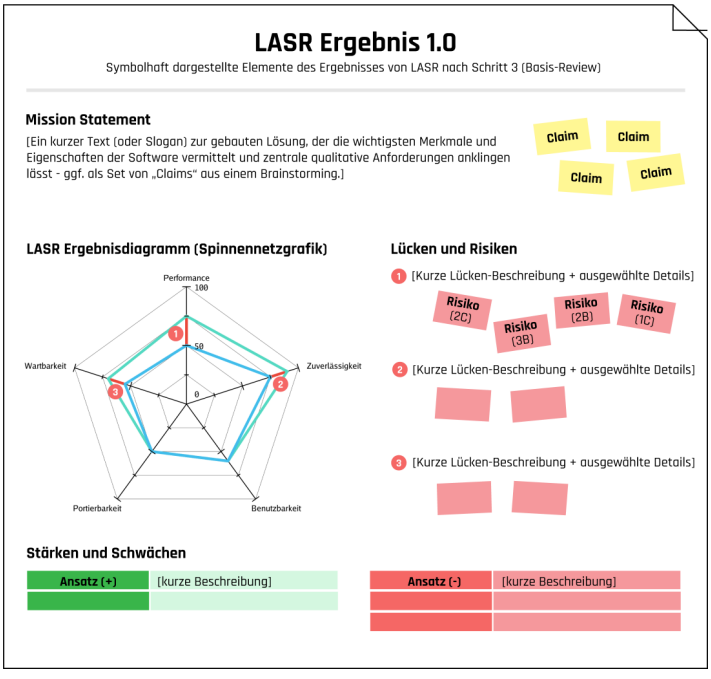


- Ziellinie
- Abweichungen (Lücken)
- Ergebnislinie

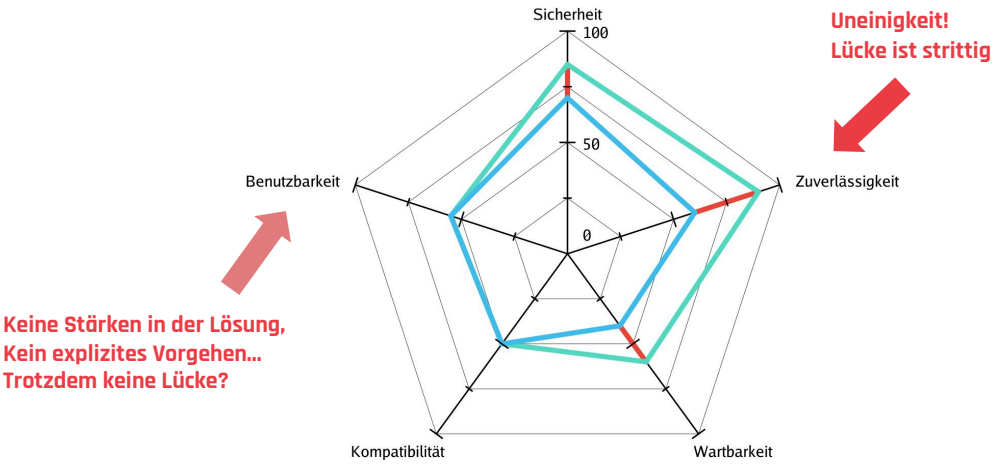
Basis-Review, 1. Ergebnis nach Schritt 3



Ergebnis 1.0



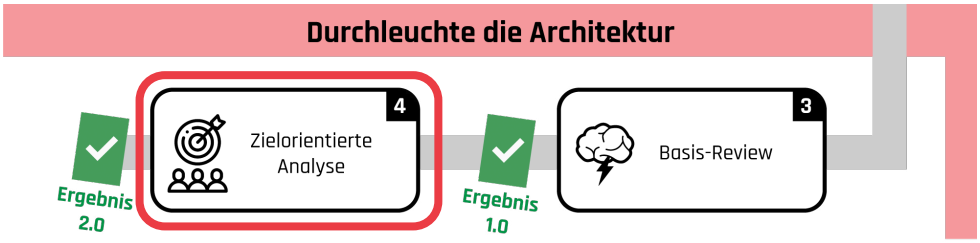
Fokus: Zielorientierte Analyse (Beispiele)



Uneinigkeit!
Lücke ist strittig

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

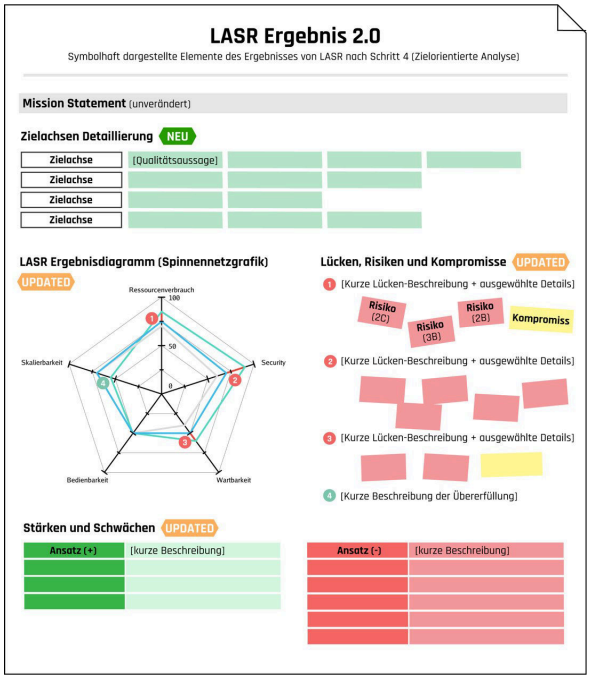
Durchleuchte die Architektur



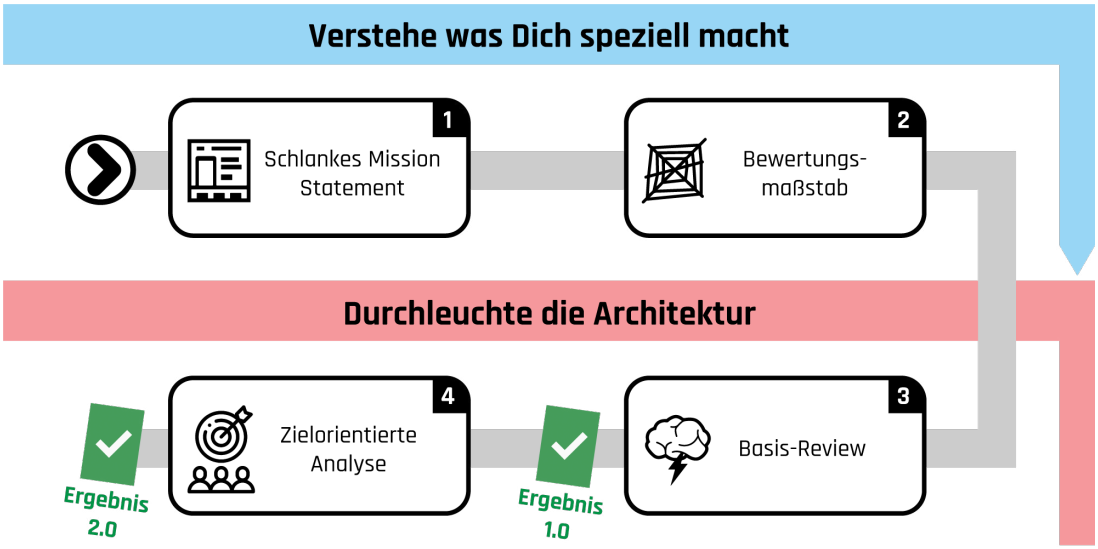
Was ist zu tun?

- 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**

Ergebnis 2.0



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.

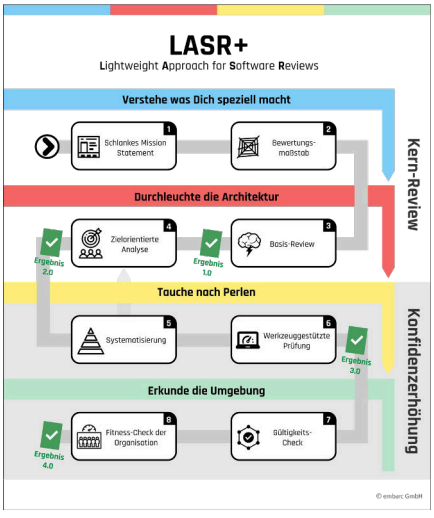
In der Theorie OK, aber was ist mit …?
Rahmenbedingungen (z.B. Standards) oder die eigene **Organisation** wurde als Risikofaktor **zu wenig betrachtet**.

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 Kern-Review …



LASR gliedert sich grob in zwei Modi, von denen der zweite optional ist („LASR+“):

I. Kern-Review

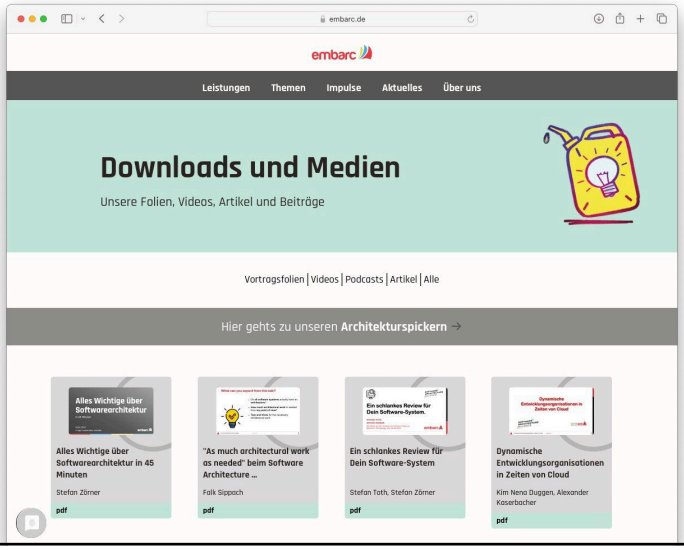
Liefert rasch ein erstes Review-Ergebnis, das im vierten Schritt bereits geschärft wird.

II. Konfidenzerhöhung (optional)

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

Weitere Informationen

Folien als PDF zum Download



➔ embarc.de/download/

Artikel zum Einstieg in LASR (online)

Stefan Toth & Stefan Zörner:
„Leichtgewichtige Software-Reviews mit LASR“
Online, Informatik Aktuell

https://www.informatik-aktuell.de/entwicklung/methoden/leichtgewichtige-software-reviews-mit-lasr.html

Buchtipp zum Thema

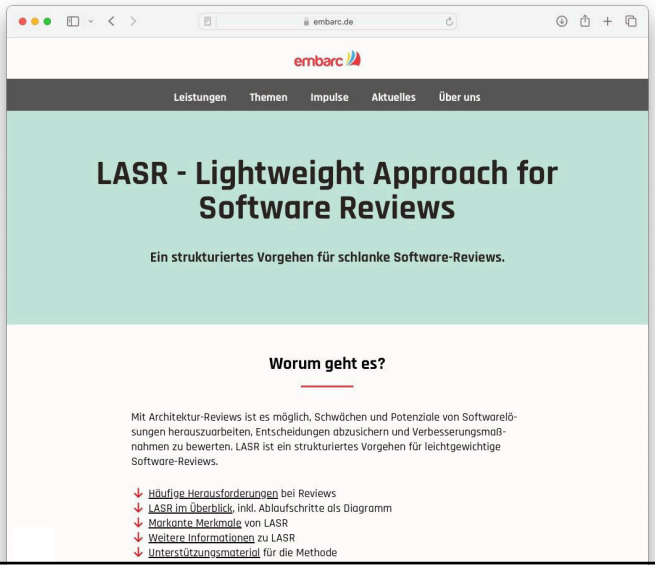
Software-Systeme reviewen
mit dem Lightweight Approach
for Software Reviews - LASR

Leanpub

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

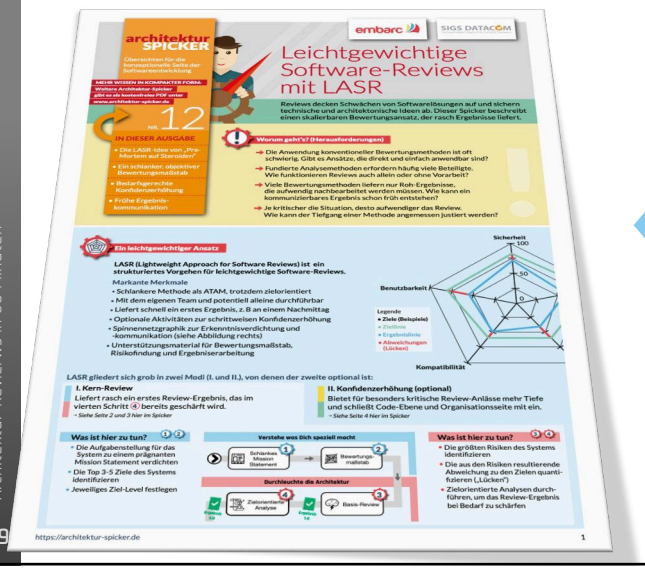
➔ leanpub.com/software-systeme-reviewen/

LASR-Themenseite bei embarc



➔ embarc.de/lasr-reviews/

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


Architektur-
Punsch
Unsere Speaker 2024
Alex Kaserbacher
Stefan Toth
Matthias Noab
Stefan Zörner
Anja Kammer
Michael Wyss
Kim Duggen
Felix Kammerlander
16. Dezember | online
Infos, Programm & Anmeldung
embarc.de/events/punsch-2024


embarc.de
Architektur-Revierws in 30 Minuten
71
Vielen Dank.
Ich freue mich auf Eure Fragen!
Stefan.Zoerner@embarc.de
linkedin.com/in/stefan-zoerner
@StefanZoerner@mastodon.social
embarc