Handeln statt jammern:
Mit REQ4ARC systematisch zu besseren Anforderungen

Peter Hruschka und Gernot Starke

In vielen Beratungsprojekten der letzten Jahre haben wir erlebt, dass sich Entwicklungsteams über zwei Dinge[1] beschweren: dass sie unter einem Mangel an guten Anforderungen leiden bzw. dass sie wichtige Anforderungen nicht oder viel zu spät erhalten haben. Sie schieben die Schuld dann auf Requirements Engineers, Business Analysts oder Product Owner, die „ihren Job nicht gut gemacht haben". Daraus resultieren dann Architektur- und Entwurfsentscheidungen auf der Basis impliziter Annahmen oder Vermutungen – ganz schlecht für's System oder Produkt.

Dem iSAQB-Advanced Modul „REQ4ARC" (Requirements for Architects) liegt die Idee zugrunde, dass sich Entwicklungsteams sich in puncto Anforderungen selbst helfen sollen, statt weiterhin über schlechte Anforderungen zu jammern. In anderen Worten: Als Architekt:in oder Entwickler:in sollten Sie so viel über Anforderungen lernen, dass Sie Projekte und Produkte erfolgreich gestalten können.

Falls Sie übrigens mehr für Grafiken als für Fließtext übrighaben – Sie könnten ganz ans Ende dieses Beitrags springen und eine visuelle Zusammenfassung (der großartigen Lisa Moritz, siehe [teapot418]) genießen. Wir freuen uns natürlich, wenn Sie danach wieder hier zum Text zurückkehren...

Architekturelevante Anforderungen

Bereits mit sehr wenig Zeitaufwand können Entwicklungsteams die entscheidenden (sprich: architekturrelevanten) Anforderungen klären. Starten wir beim Fundament:
Drei Punkte sollten jedem Team explizit klar sein: Die Projektziele, die Stakeholder und der Scope des Projekts. In anderen Worten: „Wo wollen wir hin?", „Wer darf (oder muss) mitspielen", und „Ist das unser Thema oder nicht" (siehe Abbildung 1).

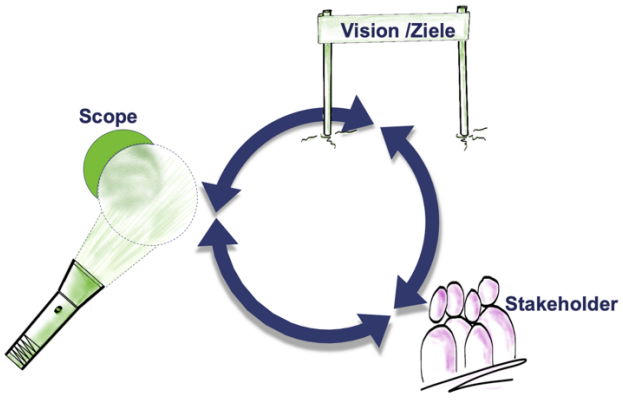


Abbildung 1: Sauber starten

[1] Klar, Teams beschweren sich auch über zu kleine Hardware, zu langsames Internet, schlechten Kaffee und die Kantine – aber *diese* Themen umschiffen wir hier.

Wenn die richtigen Personen zusammenkommen und sich die Politik aus der Diskussion heraushält, genügen dafür wenige Stunden oder - bei sehr großen Projekten - wenige Tage, um in diesen drei Kernpunkten Klarheit zu gewinnen.

Zusätzlich gehört ein Überblick über die gewünschte Funktionalität (keine Details!) dazu, sowie die wesentlichen (und wir meinen tatsächlich nur 3 – 5) Qualitätsanforderungen sowie die härtesten Randbedingungen. Unserer Erfahrung nach können Sie das in 1 – 2 Prozent des Projektaufwands ermitteln. Bei einem typischen Scrum-Team von 5 – 9 Personen und einer Laufzeit von einem Jahr (also 1000 bis 1800 Personentage), um ein Produkt iterativ, inkrementell zu entwickeln, macht das das 10 – 18 Personentage. Schon mit diesem sehr überschaubaren Zeitaufwand, können Sie das Risiko von Fehlentwicklungen deutlich abmildern.

Kern der Sache: Prozesse und Funktionen
Funktionale Anforderungen bilden ein Kernthema der gesamten Anforderungsanalyse: Es geht darum, was genau das System erledigen soll, welche Geschäfts- oder Arbeitsprozesse es unterstützen muss.
Dabei genügt es zu Beginn der Entwicklung, einen groben Überblick zu besitzen, und erst on-demand im Laufe von Entwicklungsiterationen funktionale Details zu klären. In der agilen Sprechweise beginnt diese Hierarchie bei den Epics und geht in den Stories dann ins Detail. Für Sie als Architekt:in ist diese Hierarchie deshalb wichtig, weil Sie lediglich diejenigen Teile der funktionalen Anforderungen in detaillierter Form kennen müssen, die Sie demnächst entwickeln wollen. Für andere Teile reicht Ihnen der Überblick. Das vermeidet Up-Front Arbeitsaufwand und erlaubt es, auf geänderte fachliche Rahmenbedingungen kurzfristig eingehen zu können.

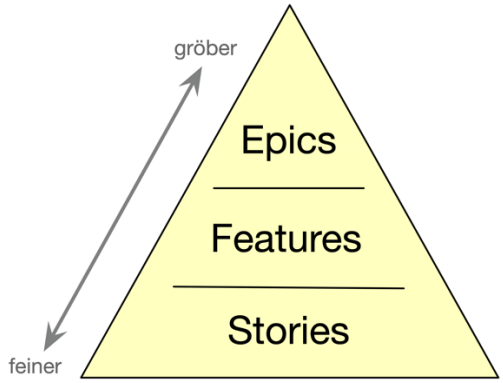


Abbildung 2: Hierarchie funktionaler Anforderungen

REQ4ARC stellt hierzu einige Mechanismen zur Hierarchisierung von Anforderungen vor, um systematisch vom „Großen“ in die notwendigen Details abtauchen zu können. In manchen Fällen tragen Techniken wie *behaviour-driven development* (BDD) oder *specification-by-example* (siehe [bdd] und [spec-by-example]) dazu bei, solche detaillierten Anforderungen mitsamt passender Abnahmekriterien entwicklungsnah und pragmatisch zu erarbeiten.

Baustelle Qualität
Die unserer Ansicht nach größte Baustelle bezüglich Anforderungen bilden mangelnde oder fehlende Qualitätsanforderungen:

Alle erwarten vom Team qualitativ gute Lösungen, aber niemand sagt explizit, was genau an Performanz, Sicherheit, Robustheit, Erweiterbarkeit, etc. benötigt wird.
Obwohl unter der Aufsicht des *International Requirements Engineering Board* (IREB, https://ireb.org) in den letzten 15 Jahre über 70.000 Personen (!!) in mehr als 80 Ländern als Requirements Engineers ausgebildet und zertifiziert wurden, und auch die Scrum-Organisationen jährlich Tausende Product Owner ausbilden, bleiben Qualitätsanforderungen in der Praxis meist unsäglich schlecht formuliert beziehungsweise implizit.
Dabei bilden doch gerade die Qualitätsanforderungen unsere wesentlichen Architekturtreiber. Ohne wirklich zu verstehen, welcher Grad an Sicherheit gebraucht wird, wie sehr das System skalieren muss, welche rechtlichen Auflagen erfüllt werden müssen, bleiben zentrale Entwurfsentscheidungen oft Glückssache.
Deshalb widmet REQ4ARC diesem Thema sehr viel Aufmerksamkeit: Wir basieren die Diskussion beispielsweise auf dem bewährten ISO-Standard 25010 (siehe Abbildung 3), und gehen ausführlich auf Möglichkeiten zur Konkretisierung einzelner Qualitätsanforderungen ein, etwa mit Hilfe von Qualitätsszenarien oder vergleichbaren methodischen Ansätzen.

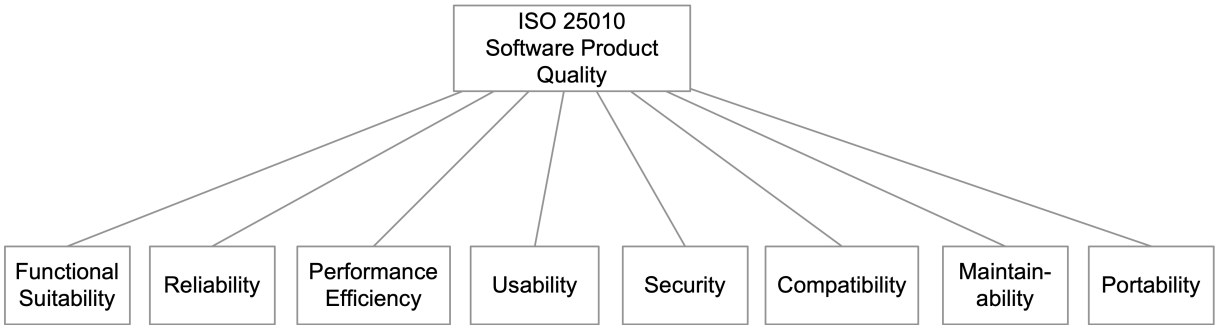


Abbildung 3: Standardisiertes Schema für Qualitätseigenschaften

Entwicklungsteams sollten hierbei besonderen Wert auf die explizite Prüfbarkeit der gestellten Qualitätsanforderungen legen, etwa durch konkrete und messbare Abnahmekriterien. REQ4ARC Trainings enthalten zu diesem Thema intensive Übungsanteile.

Enge Kooperation

Finden Sie den Fehler in folgender Aussage:

> „Kund:innen und Nutzer:innen wissen genau, was sie wollen."

Klar, ein Klassiker: Menschen (hier: unsere Fachseite) wissen eben in vielen Fällen *nicht* genau, was sie wollen oder benötigen. Daher sollten Sie für eine möglichst enge Zusammenarbeit zwischen Entwicklungsteam und den produktverantwortlichen Stakeholdern (etwa: Requirements Engineering oder Product Owner) sorgen. Uns gefällt das Motto *„discover to deliver"* der Produktexpertin Ellen Gottesdiener (siehe [gottesdiener]). Sie verknüpft die beiden Themen Requirements Analyse (Discover) und Architektur/Entwicklung/Deployment (Deliver) eng miteinander, statt sie sequentiell hintereinander auszuführen, siehe Abbildung 4.

Abbildung 4: discover to deliver

Im Rahmen dieser kontinuierlichen Zusammenarbeit entwickeln (*deliver*) Sie Teile des Produktes, und auf dieser Basis entdecken (*discover*) die fachlichen Stakeholder geänderte oder neue Anforderungen. So bekommen beide Seiten schnelles Feedback und können gemeinsam das passende System gestalten.
Alternativ dazu stellen auch Ansätze wie Three Amigo Sessions, Design Thinking oder Lean Development die Teams so auf, dass beide Fähigkeiten intensiv zusammen arbeiten und in kurzen Zyklen den Produkterfolg sicherstellen.

Zusammenfassung

In unserer pragmatischen Projektarbeit stellen wir immer wieder fest, dass Architekt:innen und Entwicklungsteams leider nur selten adäquat mit hinreichend guten Anforderungen versorgt werden. Die brauchen sie aber dringend, um Entscheidungen über die Architektur von Produkten zielgerichtet und zukunftssicher treffen zu können.
In Kurzform:

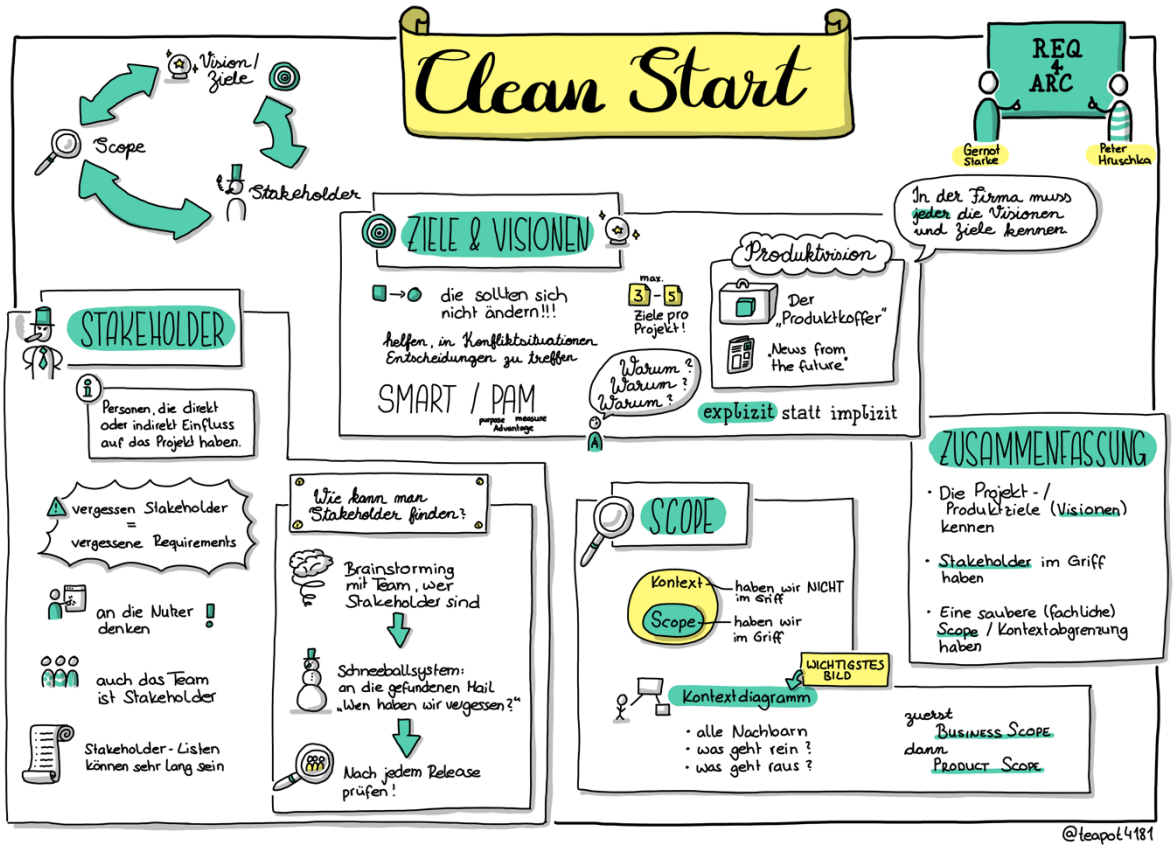
1. Klären Sie gemeinsam mit den maßgeblichen Stakeholdern Ziele, Scope und Kontext.
2. Verschaffen Sie sich einen Überblick der wichtigen High-Level Abläufe, Prozesse oder Geschäftsfunktionen (in agiler Sprechweise: Epics).
3. Klären Sie die 3-5 wichtigsten Qualitätsanforderungen, am besten mess- oder entscheidbar.
4. Im Laufe der Entwicklung verfeinern Sie (oder bitten Fachseite oder Product-Owner darum) die High-Level Prozesse on-demand in Stories oder Features.
5. Kooperieren Sie dabei mit den für das Produkt oder System verantwortlichen Personen (z.B. Fachseite, Business, PO).

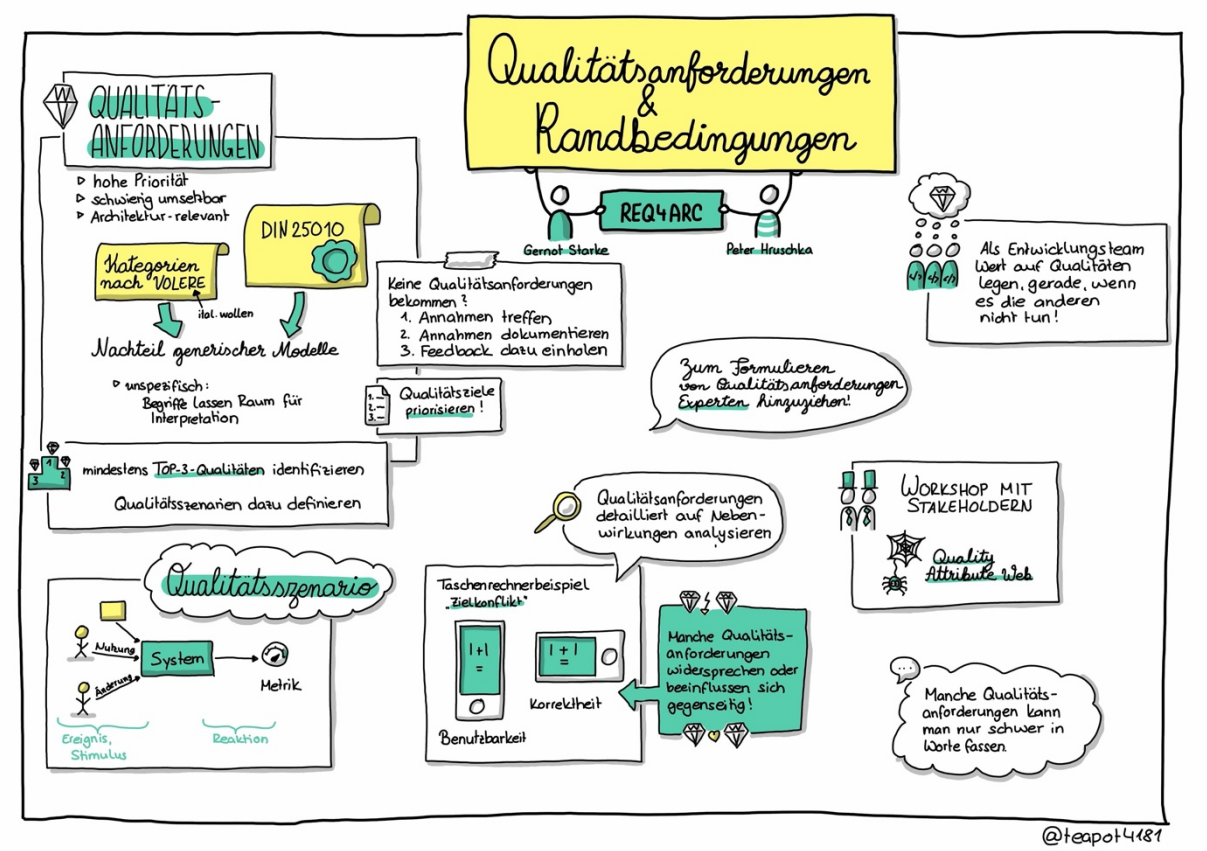
Als Hilfe zur Selbsthilfe gibt es mit dem praxisnahen REQ4ARC Modul eine effektive Abhilfe, die Ihnen Anregungen und konkrete Hilfestellung gibt, wie Sie diese Situation in den Griff bekommen. Möge die Macht guter Requirements mit Ihnen sein.

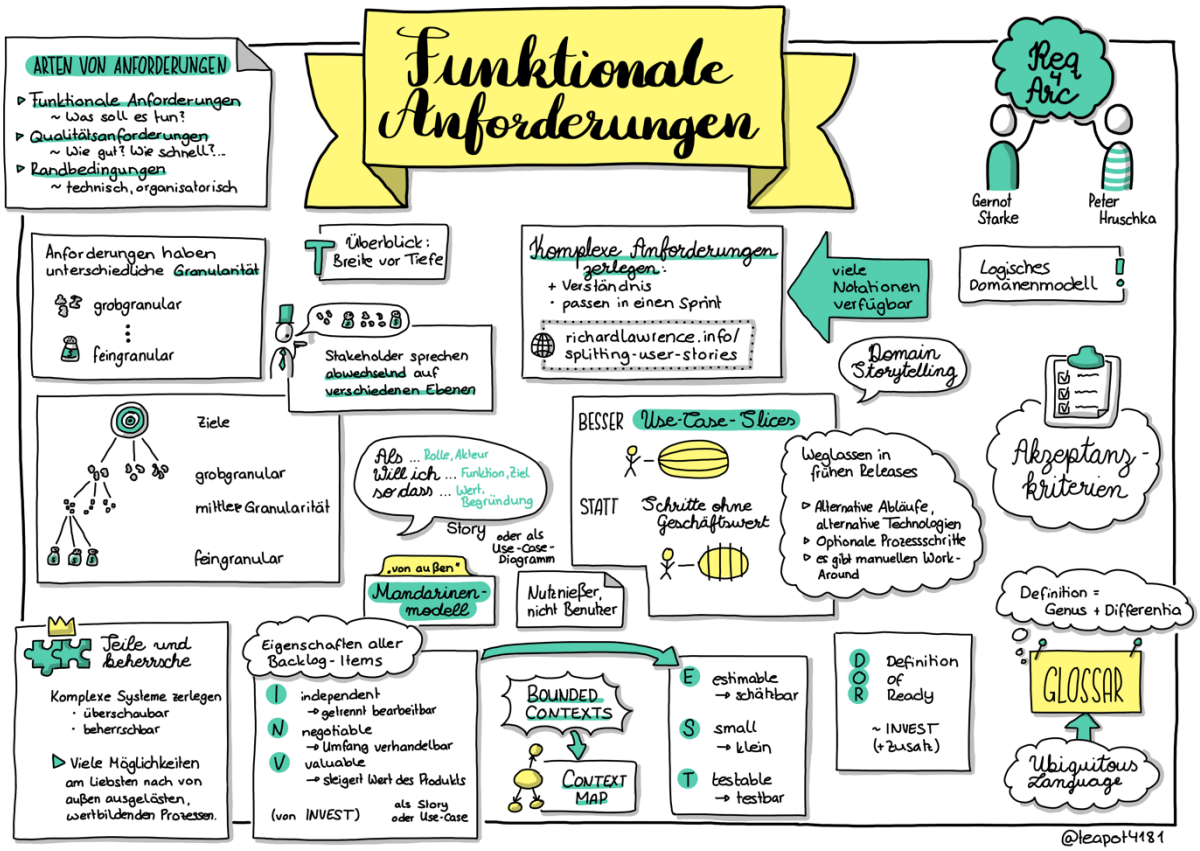
Req4Arc als Sketchnotes

Wie in der Einleitung angekündigt finden Sie hier eine grafische Zusammenfassung des Themas, erstellt von Lisa Moritz ([teapot418]). Lisa hat an einem unserer Req4Arc Trainings teilgenommen – die folgenden Abbildungen stellen ihre Mitschrift dar (falls Sie davon auch so begeistert sind wie wir, schauen Sie auf ihre Website https://www.sketchnotes.tech.

(hier die drei Sketchnotes Abb-5, Abb-6 und Abb-7 hintereinander)


Clean Start
REQ + ARC
Gernot Starke
Peter Hruschka
Vision / Ziele
Scope
Stakeholder
ZIELE & VISIONEN
die sollten sich nicht ändern!!!
max. 3-5 Ziele pro Projekt!
helfen, in Konfliktsituationen Entscheidungen zu treffen
SMART / PAM
purpose measure Advantage
Produktvision
Der „Produktkoffer"
„News from the future"
In der Firma muss jeder die Visionen und Ziele kennen.
Warum? Warum? Warum?
explizit statt implizit
STAKEHOLDER
Personen, die direkt oder indirekt Einfluss auf das Projekt haben.
vergessen Stakeholder = vergessene Requirements
an die Nutzer ! denken
auch das Team ist Stakeholder
Stakeholder-Listen können sehr lang sein
Wie kann man Stakeholder finden?
Brainstorming mit Team, wer Stakeholder sind
Schneeballsystem: an die gefundenen Mail „Wen haben wir vergessen?"
Nach jedem Release prüfen!
SCOPE
Kontext
Scope
haben wir NICHT im Griff
haben wir im Griff
WICHTIGSTES BILD
Kontextdiagramm
• alle Nachbarn
• was geht rein?
• was geht raus?
zuerst Business Scope
dann Product Scope
ZUSAMMENFASSUNG
• Die Projekt-/Produktziele (Visionen) kennen
• Stakeholder im Griff haben
• Eine saubere (fachliche) Scope / Kontextabgrenzung haben
@teapot4181


Qualitätsanforderungen & Randbedingungen
REQ4ARC
Gernot Starke
Peter Hruschka
QUALITÄTS-ANFORDERUNGEN
▷ hohe Priorität
▷ schwierig umsetzbar
▷ Architektur-relevant
Kategorien nach VOLERE
ideal wollen
DIN 25010
Nachteil generischer Modelle
▷ unspezifisch: Begriffe lassen Raum für Interpretation
Keine Qualitätsanforderungen bekommen?
1. Annahmen treffen
2. Annahmen dokumentieren
3. Feedback dazu einholen
Qualitätsziele priorisieren!
mindestens Top-3-Qualitäten identifizieren
Qualitätsszenarien dazu definieren
Als Entwicklungsteam Wert auf Qualitäten legen, gerade, wenn es die anderen nicht tun!
Zum Formulieren von Qualitätsanforderungen Experten hinzuziehen!
Qualitätsszenario
Nutzung
Änderung
System
Metrik
Ereignis, Stimulus
Reaktion
Qualitätsanforderungen detailliert auf Nebenwirkungen analysieren
Taschenrechnerbeispiel „Zielkonflikt"
Benutzbarkeit
Korrektheit
Manche Qualitätsanforderungen widersprechen oder beeinflussen sich gegenseitig!
WORKSHOP MIT STAKEHOLDERN
Quality Attribute Web
Manche Qualitätsanforderungen kann man nur schwer in Worte fassen.
@teapot4181



Quellen

[REQ4ARC Lehrplan] (Internationale Version) https://isaqb-org.github.io/curriculum-req4arc/

[REQ4ARC Buch] P.Hruschka & G.Starke: Requirements Skills erfolgreicher Softwareteams. Das Buch zum REQ4ARC Lehrplan. Für Leser:innen des iSAQB Blog 50% Ermässigung: https://leanpub.com/requirements-skills/c/isaqb-blog-coupon

[bdd] Behaviour Driven Development, siehe etwa https://cucumber.io/docs/bdd/

[spec-by-example] Gojko Adzic: Specification by Example, How Successful Teams Deliver the Right Software. Manning, 2011

[teapot4181] Lisa Moritz, https://www.sketchnotes.tech/about/

[gottesdiener] Ellen Gottesdiener, Discover to Deliver. https://www.ebgconsulting.com/agile-resources/agile-books/