SEQIS

Ten more things

REVolution – Wie ergänzen sich Anforderungsanalyse und Agile?

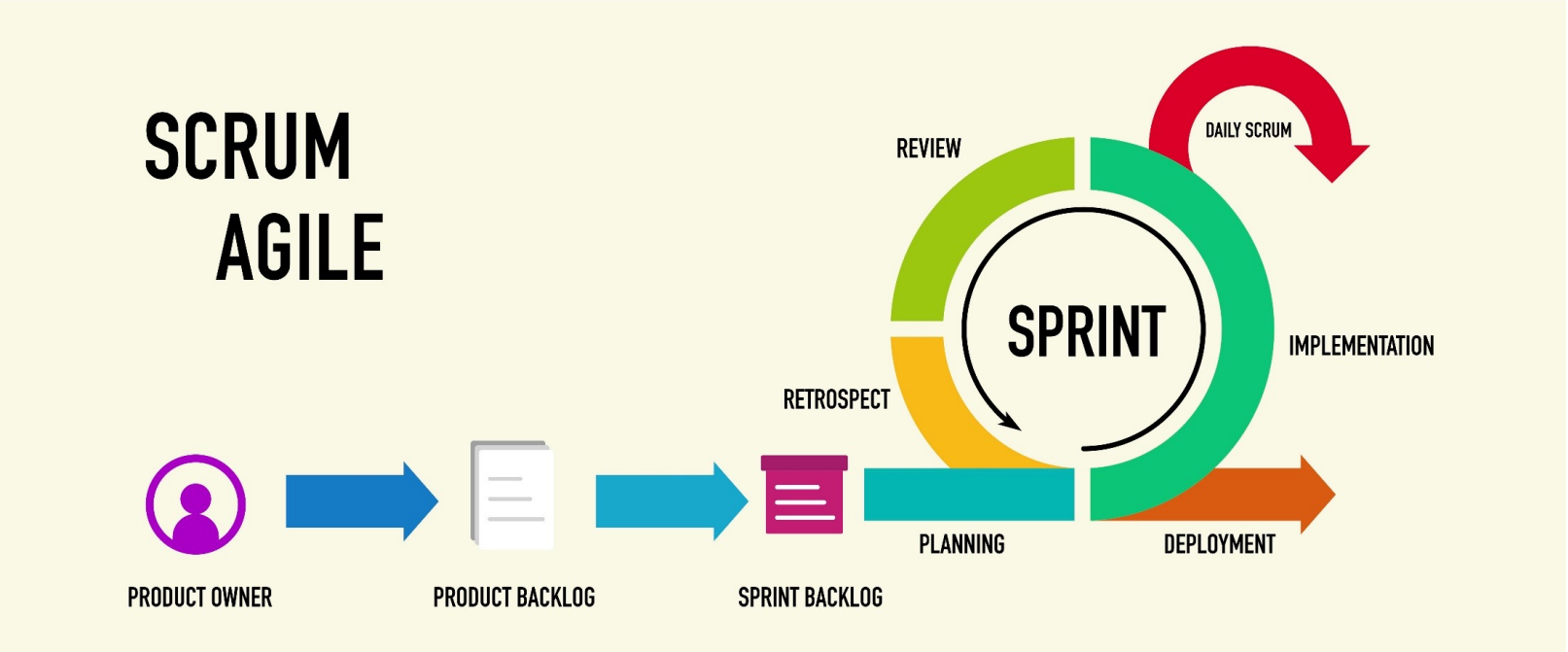
Andreas Steiner

Was ist Requirements Engineering?

- Requirements
  - Anforderungen, Voraussetzungen, Notwendigkeiten, Bedingungen
- Engineering
  - technisch, ingenieurwissenschaftlich, technische Planung

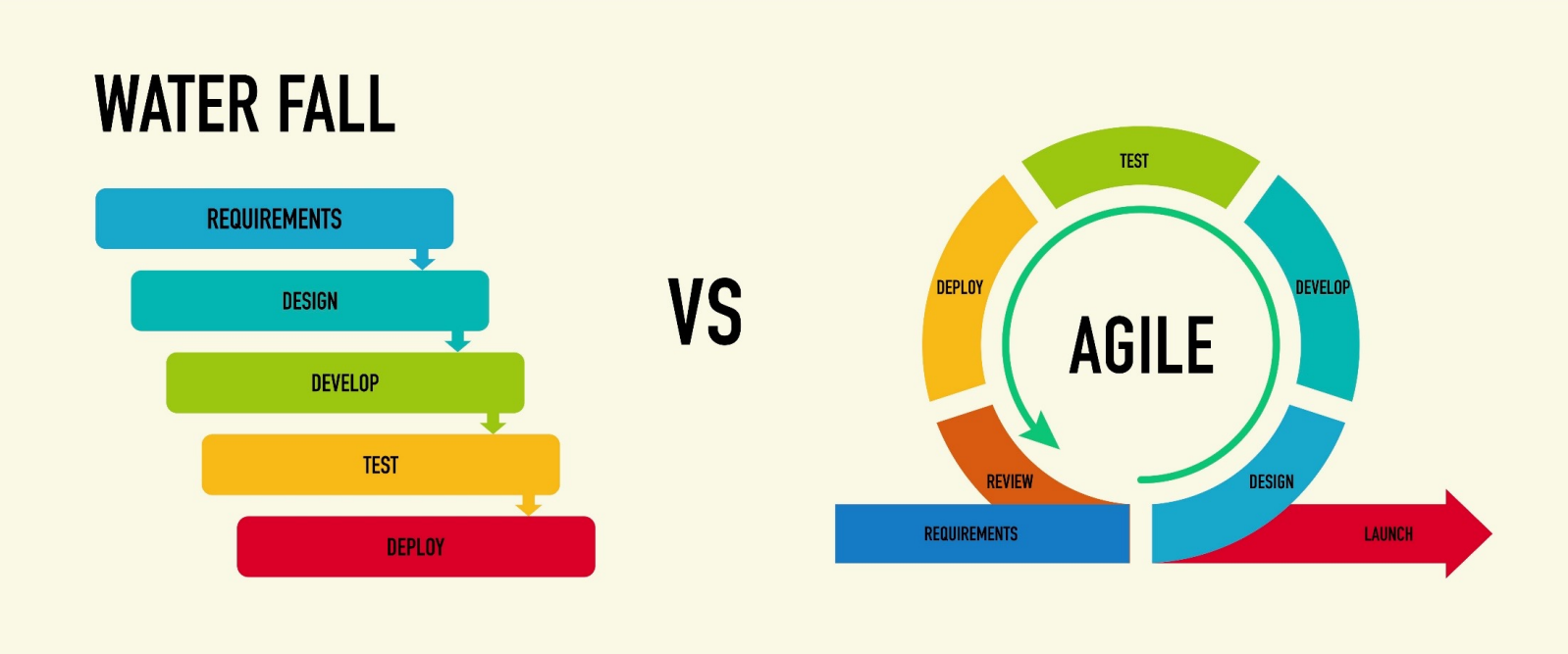
Requirements Engineering ↔ Anforderungsmanagement

Requirements Engineering heute…



iStock.com/iam2mai

Linear vs. Agil



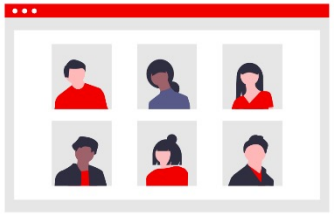
iStock.com/iam2mai

1. Die Aufgaben im Requirements Engineering

- Werden Sie sich bewusst, welcher Aufwand mit Requirements Engineering in Verbindung steht
- Gutes Requirements Engineering passiert nicht „nebenbei“

Requirements Engineering ist…

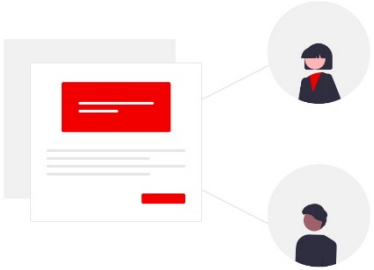
1. Anforderungserhebung
   - Ermitteln
   - Analysieren
   - Spezifizieren

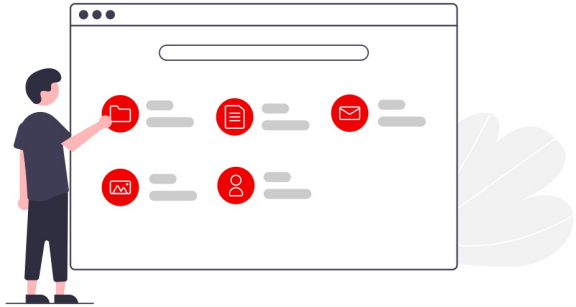


2. Anforderungsdokumentation
   - Dokumentieren
   - Zielgruppe der Dokumentation
   - Instandhaltung der Dokumentation

Requirements Engineering ist…

3. Anforderungsprüfung und -abstimmung
   - Abgestimmt
   - INVEST-Prinzip
4. Anforderungsmanagement
   - Verwendete Tools
   - Versionierung
   - Priorisierung





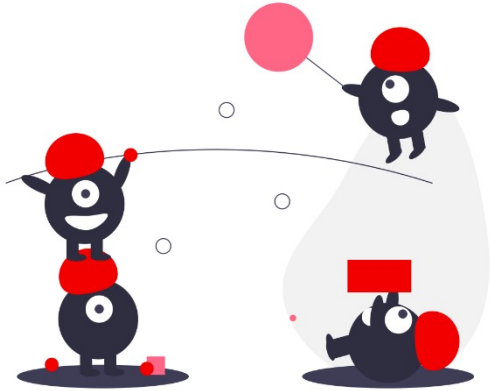
2. Räumen Sie dem Requirements Engineering die Zeit ein, die dafür benötigt wird

- Unterstützen Sie Ihre/n Product Owner/in!
- Requirements Engineering benötigt Zeit und Kommunikation

Das PO-Dasein in der Praxis

- Oftmals vom Fachbereich besetzt, daher wenig Kenntnisse im RE
- Teilzeit-Product Owner
  - User Stories müssen als „just enough“-Doku reichen
  - Schwer erreichbar
- Proxy-Product Owner
  - Darf nicht selbst entscheiden
- Visionäre Product Owner
  - Beschäftigt sich mit zukünftigen Features

Wie können wir damit umgehen?



- Agile Master
- Entwicklungsteam
- „Berater“ des PO

3. Bewahren Sie den Überblick

- Komplexität sichtbar machen
- Alle Kompetenzen ins Team delegieren

Detaillierungsgrade der Dokumentation

Kundensicht:

- Visionen/Ziele
- Epics/Stories
- Use Cases und Szenarien

Managementsicht:

- Team und Projektplan
- Releasepläne
- Storymaps
- Taskboard

Entwicklungssicht:

- Technische Vorgaben
- Kontextbeschreibungen
- Softwarearchitektur

So behalten Sie den Überblick!

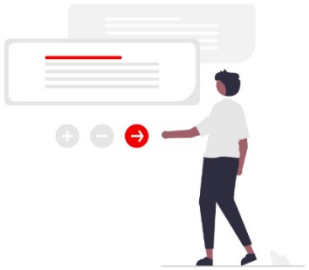
- Dokumentieren Sie ausreichend aber nicht zu viel
- Bauen Sie Kompetenzen im Team auf
- Komplexität benötigt mehr Dokumentationsaufwand

4. Priorisieren Sie User Stories und lassen Sie diese schätzen

- Priorisierung ist wichtig, um zu wissen, was als nächstes getan werden muss
- Aufwandschätzung bietet Ihnen mehr als nur Planungssicherheit

Wir priorisieren um…

- … eine Grundlage für die „just enough“-Dokumentation zu erhalten
- … uns auf die kritischen Anforderungen zu fokussieren
- … Ressourcen zu sparen
- … trotzdem ein hohes Maß an Qualität zu liefern



Die Aufwandsschätzung

- Mehr als nur Planungssicherheit
- Umsetzung einer Story muss innerhalb eines Sprint möglich sein
- Aufwandsschätzung als Tool zur Förderung der Kommunikation

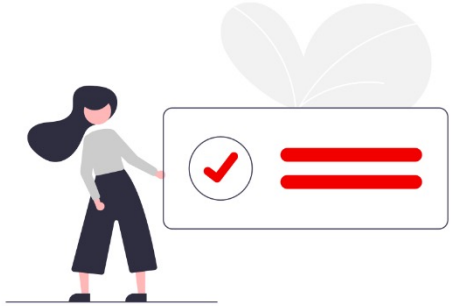
- Methoden
  - Magic Estimation, T-Shirt Sizing, Planning Poker

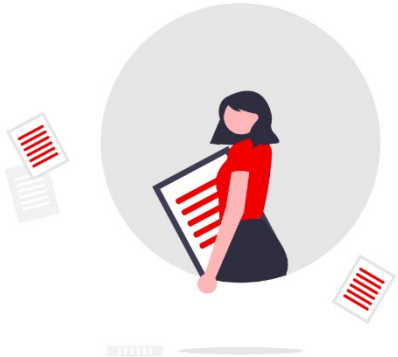
5. Dokumentieren Sie angemessen

- Beginnen Sie mit groben Anforderungen
- Und verfeinern diese bis zum Sprint Planning

Dokumentieren Sie angemessen…

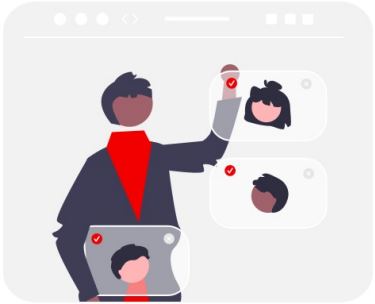
- Epics
- User Stories
- Use Cases
- Use Case Szenarien





Zuerst grob…

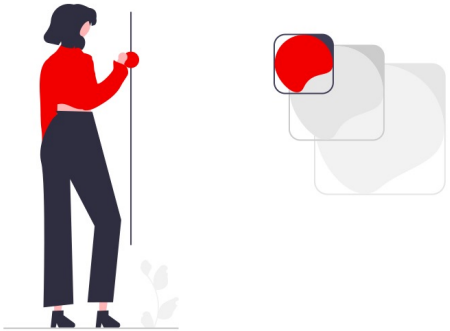
- Epics
  - Große übergeordnete Anforderung
  - Ausschließlich inhaltliche Aspekte
  - Unterhalb der Ziele und Vision
  - Oberhalb der User Stories
  - Schwerpunkt der Erstellung befindet sich am Anfang des Projekts



Beispiel:
Die Zeiterfassung wird effiziente Erfassungs- und Auswertungsmöglichkeiten für alle Zeitarten (Tagesarbeitszeit, Fehlzeiten etc.) bereitstellen.

Konkretisieren…

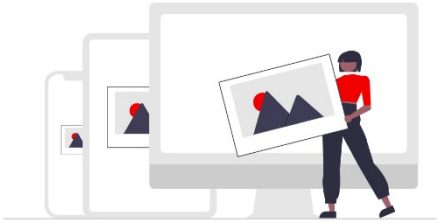
- User Stories
  - Kurze funktionale Beschreibung aus Benutzersicht
  - Ausreichender Detaillierungsgrad für die Planung der nächsten Iteration



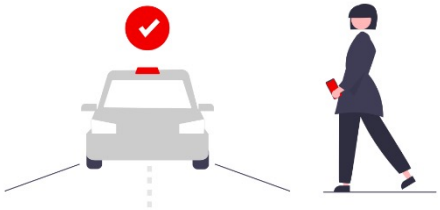
Schablone:

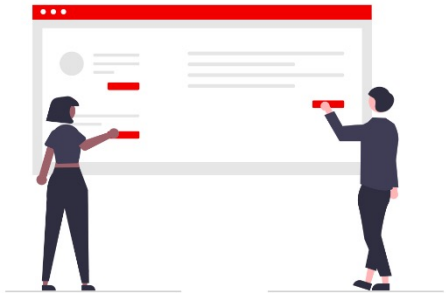
- Als <Benutzer/Systemrolle>
- Will ich <Aktion>
- Sodass/weil <Grund für das Ziel/Nutzen>

Verfeinern…

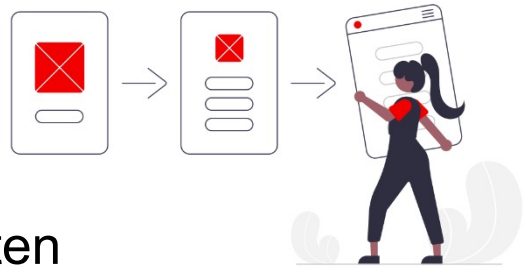


- Use Cases „Anwendungsfälle“
  - Beschreiben die Interaktion mit dem System
  - bzw. das Verhalten des Systems aus Benutzersicht
  - Grundlage für Use Case Beschreibungen und Szenarien
  - Ebene parallel zu User Stories aber auch unterhalb je nach Grad der Detaillierung





Detaillierter Ablauf…



- Use Case Szenarien
  - Spezifischer Weg durch einen Use Case
  - Stellen Anforderungen an das Systemverhalten
  - Stellen Zusammenhänge zwischen Requirements dar

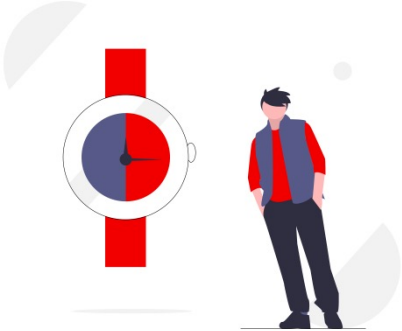
Aktivitätsdiagramme helfen bei der Darstellung und dabei den **Überblick** zu **behalten**!

6. Dokumentieren Sie rechtzeitig

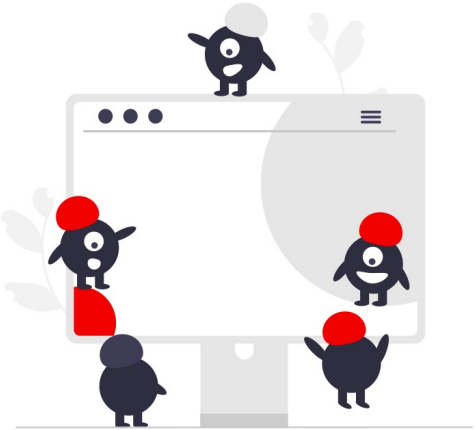
- Detaillierung bevor die Programmierung beginnt
- Die Dokumentation unterstützt eine gute Programmierung

Dokumentieren Sie rechtzeitig!

- Die Dokumentation ist vor der Umsetzungsiteration abzuschließen!
- Die Verfeinerung der Use Cases, Use Case Szenarien und User Stories muss vor der Programmierung abgeschlossen sein!
  - Deadline: Sprint Planning



Was bisher geschah…



- Epics erhoben
- User Stories abgeleitet
- Use Cases beschrieben
- Use Case Szenarien erstellt
- Anforderungsdokumentation auf die erforderlichen Qualitätskriterien überprüft

Also ist das Team bereit für die Umsetzung oder?

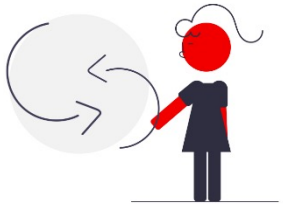
7. Definition of Ready

- Erstellen Sie eine Definition of Ready
- Passen Sie die Definition of Ready bei Bedarf vor dem nächsten Sprint an

Das Quality Gate für die agile Entwicklung!

- Wird vor Beginn des agilen Prozesses definiert
- Legt fest welche Quality Gates ein Arbeitspaket erfüllen muss, bevor es in den Sprint Backlog aufgenommen werden darf.

Nicht vergessen: Die **Definition of Ready** darf und **soll** bei Bedarf und vor dem nächsten Sprint angepasst werden!



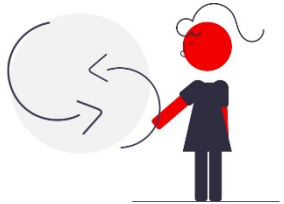
8. Die Definition of Done

- Erstellen Sie eine Definition of Done
- Passen Sie die Definition of Done bei Bedarf vor dem nächsten Sprint an

Das Quality Gate für ein erfolgreiches Ergebnis!

- Wird vor Beginn des agilen Prozesses definiert
- Legt fest welche Quality Gates ein Arbeitspaket erfüllen muss, bevor es abgeschlossen werden darf.

Nicht vergessen: Die **Definiton of Done** darf und **soll** bei Bedarf und vor dem nächsten Sprint angepasst werden!



9. Wagen Sie den Mut zur Lücke!

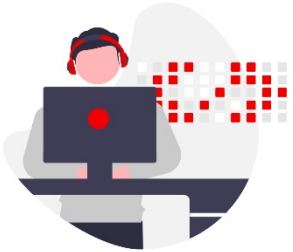
- Was just-enough wirklich bedeutet
- Begeistern Sie Ihre Kunden

Was sagt das agile Manifest?

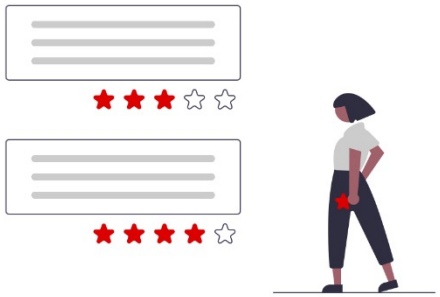
Funktionierende Software

ist **wichtiger** als

umfassende Dokumentation!



Das Kano-Modell



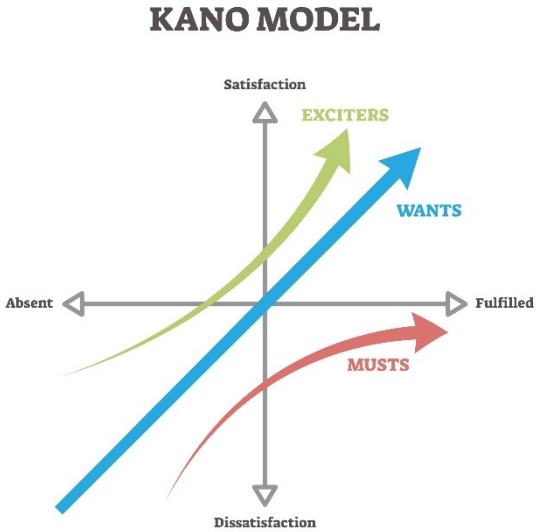
- Entwickelt von Noriaki Kano im Jahre 1970
  - Basierend auf Frederick Herzberg
  - Motivationsfaktoren vs. Hygienefaktoren

Erkenntnis:

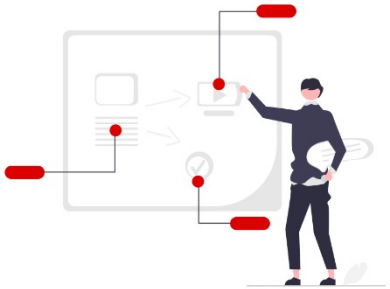
- Unzufriedenheit und Zufriedenheit sind zwei unabhängige Eigenschaften
- Keine Unzufriedenheit ≠ Zufriedenheit
- Kundenanforderungen sind unterschiedlich zu gewichten

Die 5 Faktoren

- Basisfaktoren
- Leistungsfaktoren
- Begeisterungsfaktoren
- unerhebliche Faktoren
- Rückweisungsfaktoren



iStock.com/VectorMinde



Wie funktioniert „just enough“?

Wagen Sie den Mut zur „Lücke“!

Ausbleiben von Basisfaktoren?

Best Case: Der Basisfaktor wird zu einem unerheblichen Faktor

Worst Case: Der Basisfaktor kann mit einem Update nachgereicht werden oder ist

Übertrifft ein Produkt die **Erwartungen nach dem Kauf**, entsteht **hohe Kundenzufriedenheit!**

Beispiele aus der Praxis

- iPhone 2007
  - keine Copy and Paste-Funktion
- iPad 2009
  - Keine Kamera, obwohl das iPhone 3G (2008) bereits eine Kamera hatte

Bildquellen:
Carl Berkeley from Riverside California, IPhone First Generation 8GB (3677961514), CC BY-SA 2.0
Evan-Amos, IPad-WiFi-1stGen, als gemeinfrei gekennzeichnet, Details auf Wikimedia Commons

10. Prozessverbesserung ist ein Lernprozess

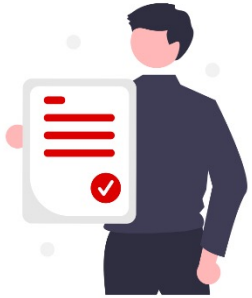
- Es ist nicht alles in Stein gemeißelt und vieles ist vom Kontext abhängig
- Veränderung benötigt Zeit, optimieren Sie Stück für Stück und nicht alles auf einmal

Bewusstmachen von Problemen

- Sprechen Sie Probleme und Unzufriedenheit an
  - Als Mitglied des Entwicklungsteams → Sprint Retro
  - Als Außenstehender → Agile Master

- Veränderung ist ein Lernprozess und benötigt
  - **Zeit und Geduld**

Fördern von Requirements Engineering



- Als Product Owner
- als Entwickler
- als Tester
- oder in jeder anderen Rolle.

Empfehlung: Requirements Engineering im Team verankern

SEQIS

Ten more things

REVolution – Wie ergänzen sich Anforderungsanalyse und Agile?

andreas.steiner@SEQIS.com

Go 4 continuous improvement