Inhalt der Vorlesung

- Einführung
- Kommunikation
- Software Qualität
- Vorgehensmodelle
- Requirements Engineering
- Konfiguration Management

Literatur

- Pohl, Rupp: Basiswissen Requirements Engineering, dpunkt.verlag, 3. Auflage, 2011
  Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level

- C.Rupp&die Sophisten: Requirements-Engineering und -Management, 6.Auflage, Hanser

- R. Wirdemann: Scrum mit User Stories, 3. Auflage, Hanser, 2017

- Larman: Practices for Scaling in Lean & Agile Development, Addison-Wesley Professional

Überblick

- RE – Rückblick/Wiederholung
- RE in Scrum Projekten

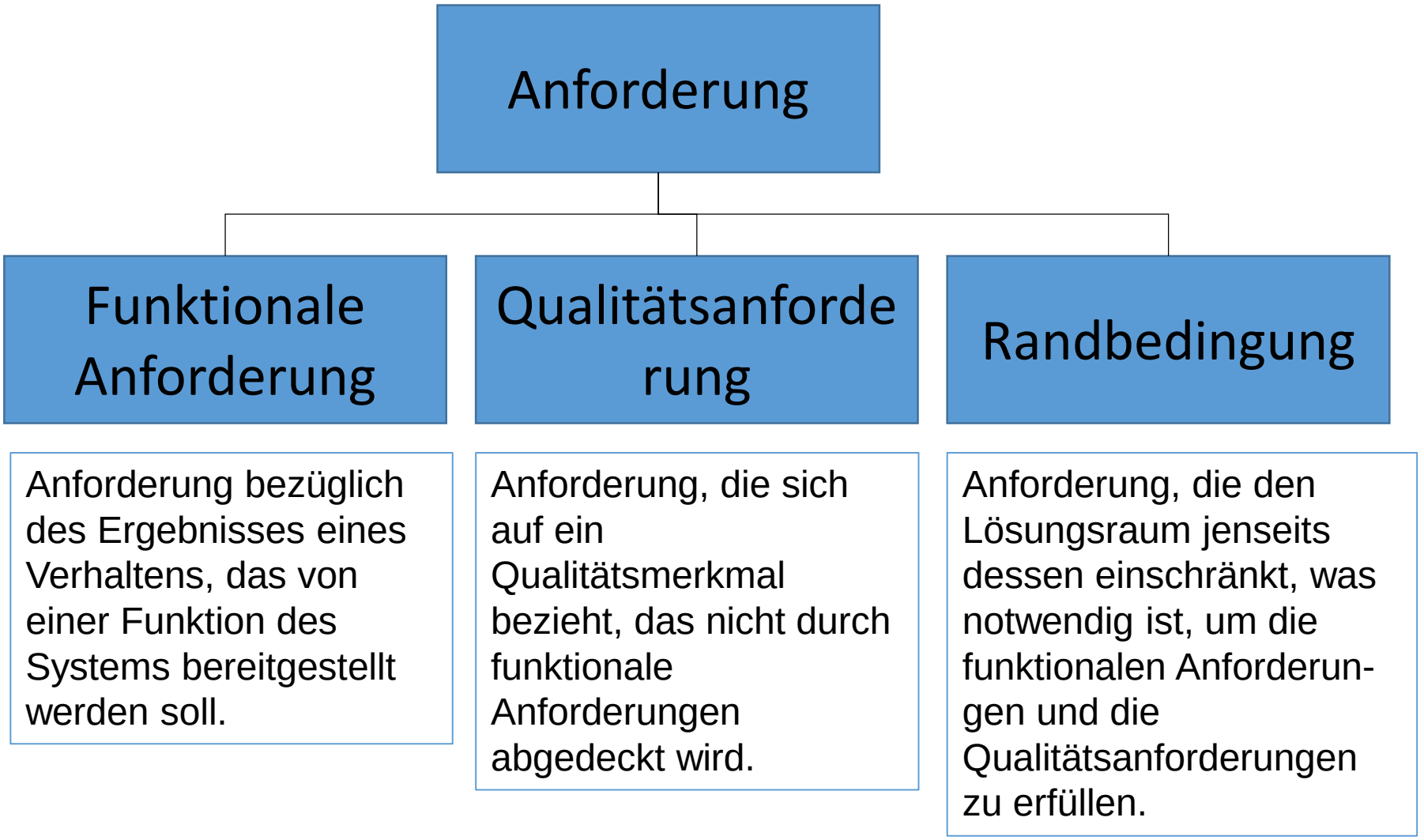
Überblick

- RE – Rückblick/Wiederholung
- RE in Scrum Projekten

Anforderungen

Anforderungen legen fest, was man von einem Softwaresystem als Eigenschaften erwartet.

Arten von Anforderungen



Definition Requirements Engineering

Das Requirements Engineering ist ein systematischer und disziplinierter Ansatz zur Spezifikation und zum Management von Anforderungen mit den folgenden Zielen:

1. Die relevanten Anforderungen zu kennen, Konsens unter den Stakeholdern über die Anforderungen herzustellen, die Anforderungen konform zu vorgegebenen Standards zu dokumentieren und die Anforderungen systematisch zu managen.
2. Die Wünsche und Bedürfnisse der Stakeholder zu verstehen, zu dokumentieren sowie die Anforderungen zu spezifizieren und zu managen, um das Risiko zu minimieren, dass das System nicht den Wünschen und Bedürfnissen der Stakeholder entspricht.

Aus: Pohl, Rupp: Basiswissen Requirements Engineering

RE bisher


Qualitätsmerkmale
Anforderungen_dokumentieren
Brainstorming_paradox
Prüfen
Brainstorming
Workshops
Stakeholder-Liste
Anf._verwalten
Befragungen
Stakeholder
Prototypen
Systemkontext
Stakeholder-Portfolio
Apprenticing
Kontextgrenze
Stakeholder-Kontrakt
Anforderungen_ermitteln
Anforderungen_abstimmen

Überblick

- RE – Rückblick/Wiederholung
- RE in Agilen Projekten

Einbettung in Vorgehensmodelle

- **Wasserfall:** Eigene Projektphase
- **Agile Prozesse:** kontinuierlicher phasenübergreifender Prozess, in die Systementwicklung integriert.

Anforderungsspezifikation in der agilen Welt.

Verwendung des Product Backlogs

➔Product Backlog Items (nicht gleich Anforderungen)

➔Product Backlog Item ist nicht vollständig, bevor es diskutiert wurde

While a product backlog can be thought of as a replacement for the requirements document of a traditional project, it is important to remember that the written part of an agile user story ("As a user, I want …") is incomplete until the discussions about that story occur.
It's often best to think of the written part as a pointer to the real requirement. User stories could point to a diagram depicting a workflow, a spreadsheet showing how to perform a calculation, or any other artifact the product owner or team desires.

Aus https://www.mountaingoatsoftware.com/agile/user-stories

Product Backlog

Original Scrum Guide:

„Das Product Backlog ist eine geordnete Liste von allem, was in dem Produkt enthalten sein kann. Es dient als einzige Anforderungsquelle für alle Änderungen am Produkt.

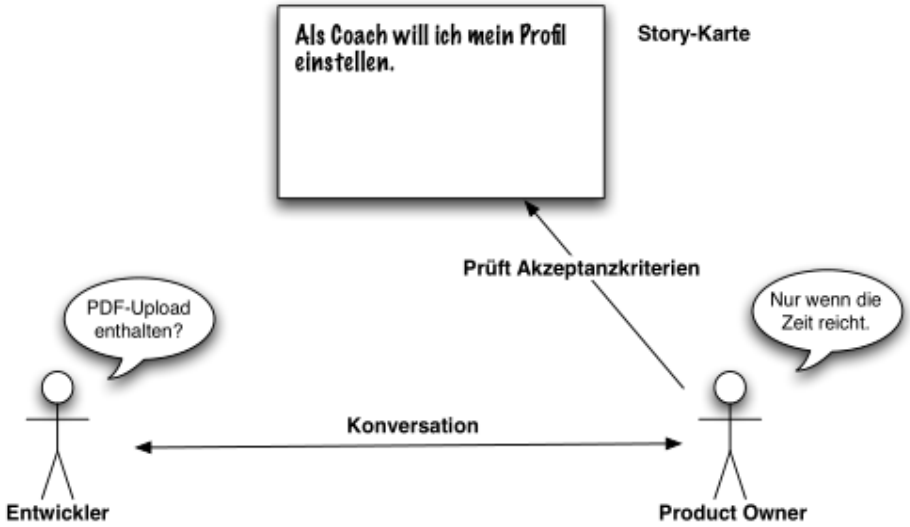
…

Im Product Backlog werden alle Features, Funktionalitäten, Verbesserungen und Fehlerbehebungen aufgelistet, die die Änderungen an dem Produkt in zukünftigen Releases ausmachen.“

Im Original keine Einschränkung auf User Stories oder dgl.

Themen, Epics, User Stories

- **User Story**
  (nach Wirdemann):
  - Karte
  - Konversation
  - Akzeptanzkriterien



- **Epic**: *„ Epics sind große User Stories, die weder vernünftig geschätzt noch innerhalb eines Sprint entwickelt werden können.“*
- **Thema:** *„EinThema ist kein eigenständiger Story-Typ, sondern eine Menge zusammengehöriger User Stories, die sich um ein bestimmtes funktionales Thema herum gruppieren.“*

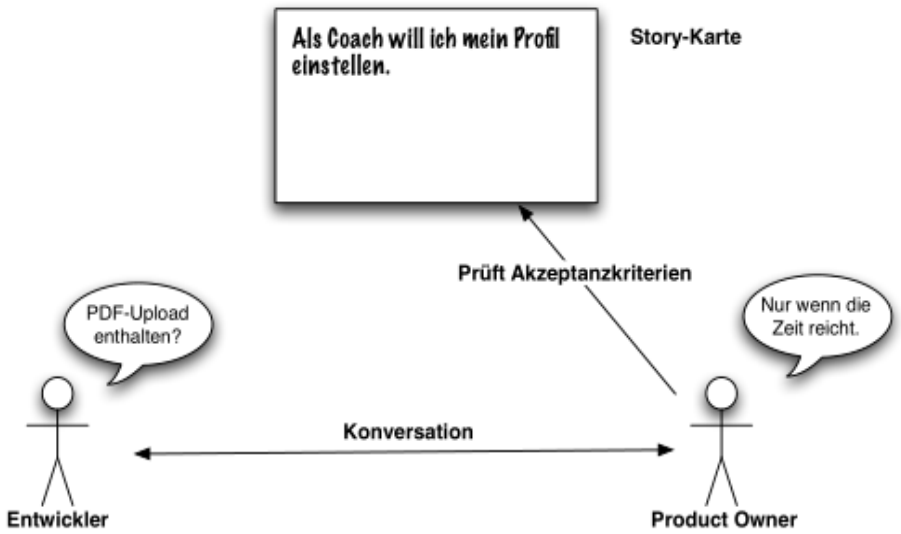
Zitate aus R. Wirdemann: Scrum mit User Stories, 3. Auflage, Hanser, 2017

Backlog Items

Product Backlog Items: Themes, Epics, User Stories

Für das Sprint Planning: User Stories

R. Wirdemann: „*Das Product Backlog ist eine Liste aller im Projekt bekannten User Stories*“



„ [die Karte] … *ist ein Versprechen des Teams an den Product Owner, sich im Detail über die Story zu unterhalten, sobald sie konkret wird. Sobald es an die konkrete Entwicklung der Story geht, beginnen die Entwickler und der Product Owner einen Dialog, der bis zur Fertigstellung der Story andauert und in vielen kleinen Feedback-Schleifen die Details der Story herausarbeitet.*“

User Stories

- In der Sprache des Kunden
- Verwendung von Benutzerrollen
- Keine Technik
- Keine Benutzeroberfläche
- Verwendung eines Sprachtemplates ➔ nächste Folie

Satzschablonen in der agilen Welt

Mike Cohn hat folgende Struktur für **User Stories** eingeführt (https://www.mountaingoatsoftware.com/agile/user-stories):

As a <type of user>, I want <some goal> so that <some reason>.

Übernommen von B. Gloger:

Als Anwender <mit der Rolle> benötige ich eine <Funktionalität>, damit ich den <Nutzen> bekomme.

Projektbeispiel

1.1.1 Generation of Business Partner reports

Field Label	Field Value	Field Label	Field Value
ReqID:	578		
Requirement Type	User Story	Priority	2-Medium

Description
As a Headquarter Staff or Manager **I want to generate reports out of the business partner database** **in order to analyze the BP System's current state**

Beispiele für User Stories

- As a power user, I can specify files or folders to backup based on file size, date created and date modified.
- As a user, I can indicate folders not to backup so that my backup drive isn't filled up with things I don't need saved.

Quelle: https://www.mountaingoatsoftware.com/agile/user-stories

Begründung für die Cohnsche Form

Siehe https://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template

1. Verwendung der Ich Form führt zu Identifikation mit der User Story.
2. Einheitliche Struktur hilft dem Product Owner bei der Priorisierung.
3. Sinnvolle Struktur (Bitte überprüfen Sie das Argument aus der Quelle oben selbst)

Schneiden von User Stories

- Vertikales Schneiden
- Schneiden nach Daten
- Schneiden nach Aufwand
- Schneiden nach Forschungsanteilen
- Schneiden nach Qualität
- Schneiden nach Benutzerrolle
- (Schneiden nach Akzeptanzkriterien)
- (Schneiden nach technischer Voraussetzung)

Siehe R. Wirdemann; Scrum mit User Stories

Und was ist mit den anderen Themen?

- Nichtfunktionale Anforderungen
- Fehler
- Refactorings
- Installationen von Bugtracking System
- …

Möglichkeiten damit umzugehen

Umformulieren, so dass der Zweck klar wird und die Anforderung in Form einer User Story dargestellt werden kann.

Hilfsmittel dazu: Zusätzliche Benutzerrollen (künstlich) einführen: Programmierer, CTO
Bsp. Aus R.Wirdemann:

Technische Anforderung	User Story
Demo-Server aufsetzen	Als Vertriebsmitarbeiter will ich das System auf einem extern zugänglichen Demo-Server präsentieren.
Datenbank Connectionpool implementieren	Bis zu 50 Scrum-Coaches wollen die Anwendung gleichzeitig benutzen.
Deployment-Prozess automatisieren	Als Programmierer will ich einen automatisierten Deployment-Prozess, so dass die Anwendung schneller und weniger fehleranfällig ausgeliefert werden kann.
Anwendung ausfallsicherer machen	Als CTO will ich, dass die Anwendung auf einem zweiten Server installiert wird, so dass das System maximal verfügbar ist.

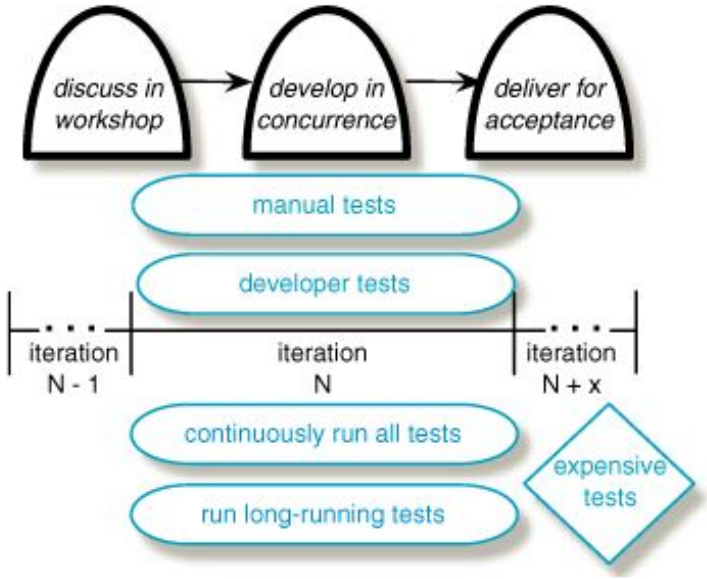
Möglichkeiten damit umzugehen

- Story-übergreifende, nicht-funktionale Anforderungen werden als Constraints formuliert.
- Fehler:
  - Sofort: Werden sofort behoben
  - Planbar: Werden in die nächsten Sprints eingeplant
- Technisches Backlog
  - Owner ist das Team (nicht der PO)
  - Einträge werden im Sprint Planning mitbearbeitet

Spezifikation mündlich vs Abnahme

Prinzip der A-TDD (Acceptance Test Driven Development)

- Tests as requirements, requirements as tests
- Workshops for clarifying requirements
- Concurrent engineering
- Prevention instead of detection



Quelle: Larman: Practices for Scaling in Lean & Agile Development, Addison-Wesley Professional