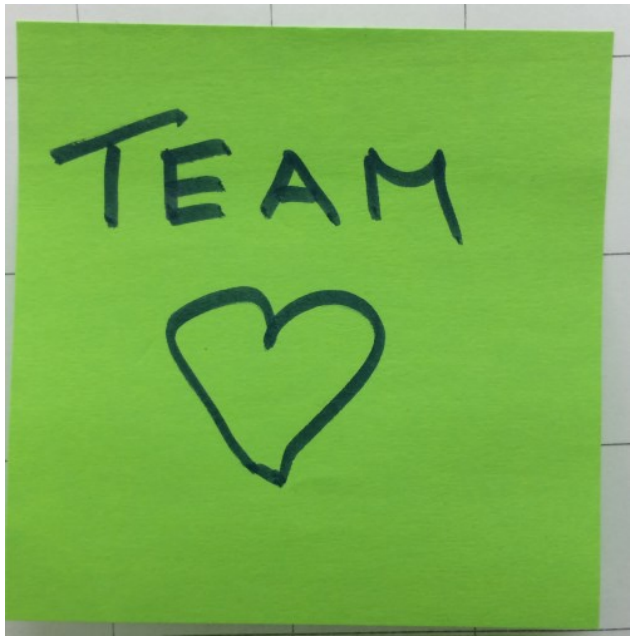
Agiles Software-Engineering

Prof. Dr. Gerd Beneken

Woche 01, Kapitel 4

Rollen



Rollen im Projekt

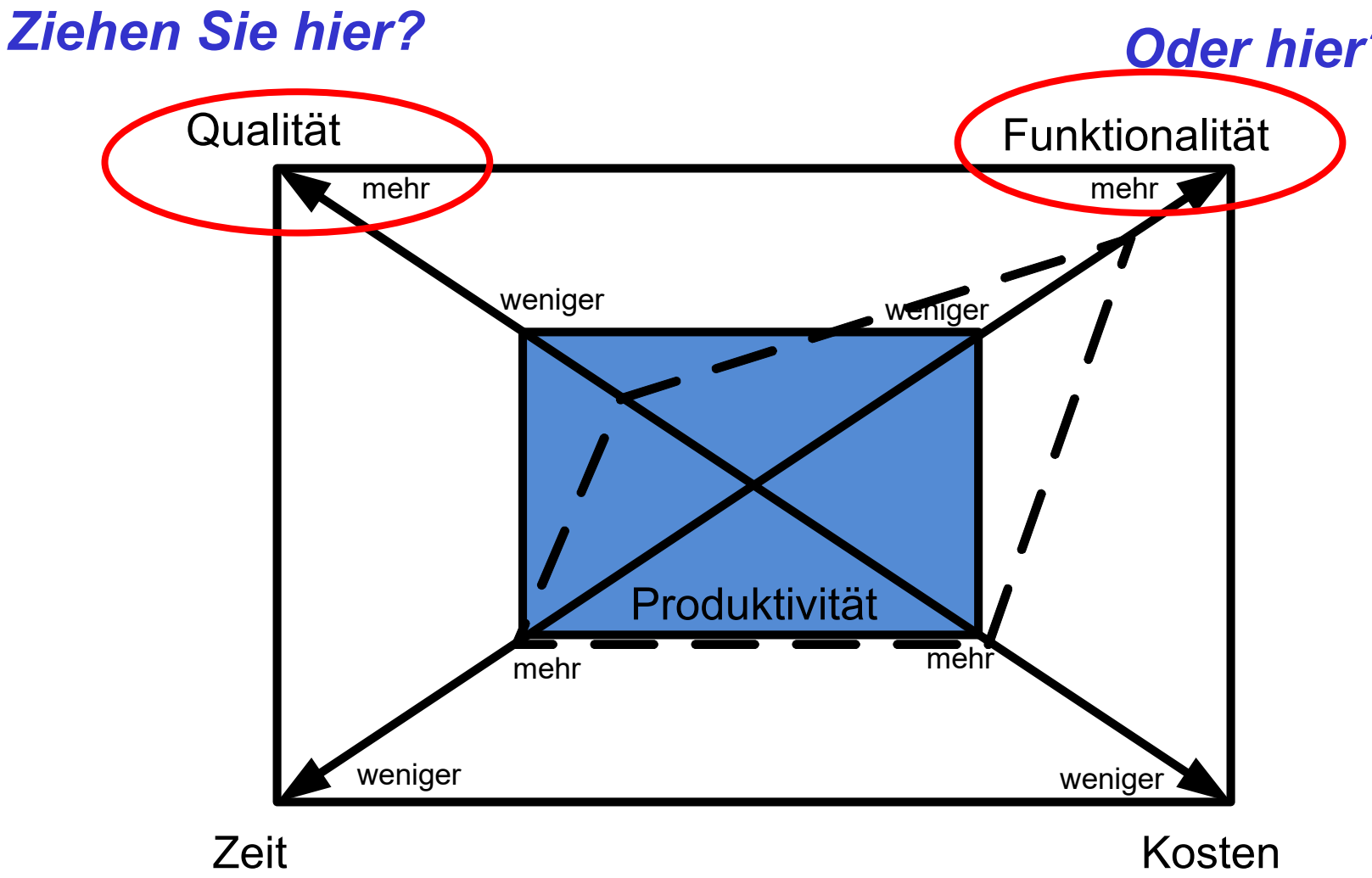
- Rolle = Wofür bin ich verantwortlich?
- ***Verantwortlich*** =
  - Termin einhalten
  - Qualität sicherstellen
  - Inhalt sicherstellen
  - ***Konsequenzen für Misslingen / Nicht Wahrnehmung tragen***
- Verantwortlich ≠ alles selber machen
- Beispiele
  - Projektleiter(in)
  - Tester(in)
  - UX/Usability-Spezialist(in)
  - Technische(r) Architekt(in) / Chefdesigner(in)
  - Requirements Engineer, Fachliche(r) Architekt(in)

Verantwortung

- Immer ***genau ein:e*** Verantwortliche:r für jedes Dokument, Ergebnis, Eigenschaft
- Grund: nur eine Person darf und muss koordinieren bzw. ändern
  - Motivation (Delegation von Verantwortung)
  - Wenn Dokument nicht aktuell, kann jemand *belangt* werden
  - Pflege des Dokuments planbar
- Verantwortliche:r darf Arbeit delegieren (an andere Autor:innen)
- Verantwortliche:r fordert Ergebnisse von Autoren ein

Teufelsquadrat nach Sneed: An welcher Ecke ziehen Sie?

Arbeiten Sie im Diskurs (Diskussion ist wichtig)



Rollen: Agil und Klassisch

Agile Rollen: Funktionsübergreifende Teams

- ***Product Owner***
- Scrum Master / Agile Coach / Catalytic Leader
- ***Teammitglied***

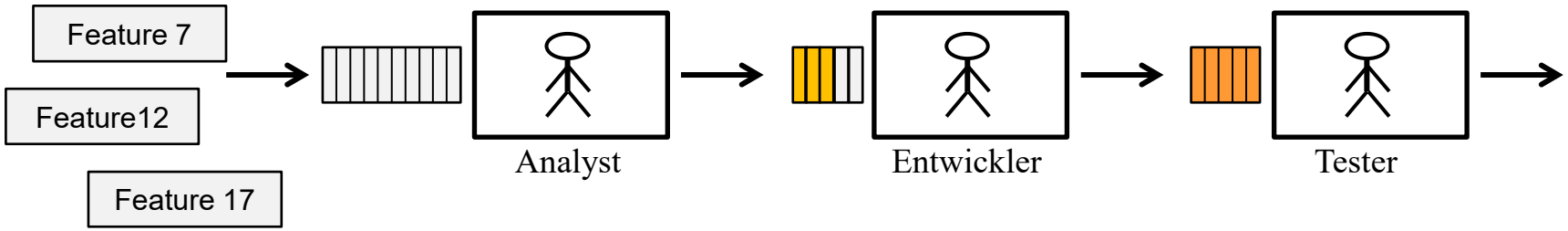
Klassische Rollen: Arbeitsteilung, Spezialisierung (Taylorismus)

- Projektleiter(in)
- Architekt(in)
- UX/Usabiltiy Spezialist(in)
- Tester(in), Qualitätsmanager(in)

Bitte Verantwortungsbereiche und ggf. auch Rollenmodell anpassen, wenn das die Projektumstände erfordern. Das obige Rollenmodell enthält einen

Agiles Rollenverständnis
Teams sind *funktionsübergreifend*

- ***Gemeinsame „Ownership“*** für das Projektergebnis
- Jede(r) ist für alles (mit) verantwortlich
- ***Spezialisierungen sind erlaubt***, aber keine Ausrede
- Dadurch schnelleres und effektiveres Arbeiten
  Grund: Keine Zwischenlager halbfertiger Features, keine Wartezeiten auf den Spezialisten, kein „Dafür ist aber ZZZ zuständig“



- Beispiel Hausbau: Verzögerungen weil ständig auf irgendeine:n Spezialist:in gewartet werden muss

Product Owner: Verantwortlich für *das Produkt*

- Verantwortung für das ***Produkt***
  - Betreuung/Anwalt der Stakeholder, Kunde fühlt sich gut betreut
  - ***Scope***: ***Anforderungen*** sind bekannt und werden gemanaged.
    Wird das gebaut, was angeboten wurde?, ***Änderungsmanagement***
  - Stakeholder: ***Sind alle relevanten Personen bekannt und beteiligt***?
- Tätigkeiten / Ergebnisse
  - Allg. Anforderungen im Backlog (Gitlab) und in Dokumenten
  - Backlog und Wiki mit dem Team zusammen pflegen
    (User Storys, GUI, Datenmodell, Schnittstellen)
  - Anforderungen abstimmen mit Kunden (Review)
  - Änderungen erkennen und managen
  - Anforderungen strukturieren

Agile Coach / Scrum Master = „Catalytic Leader“

- Verantwortung für die ***Produktivität***, den Prozess
  - ***Produktivität des Teams***
  - Hindernisse (Impediments) zusammen mit dem Team entf.
  - Probleme mit Root-Cause-Analysis verstehen
  - Team entwickeln, Einarbeitung neuer Mitarbeiter (Sensei)
  - genchi genbutsu = Management vor Ort
- Ergebnisse
  - Leitet / Moderiert Meetings
  - Achtet auf Einhaltung der Regeln
- ***Keine Weisungsbefugnis*** (servant leader), nur Überzeugung
  - ***Ask the Team!***

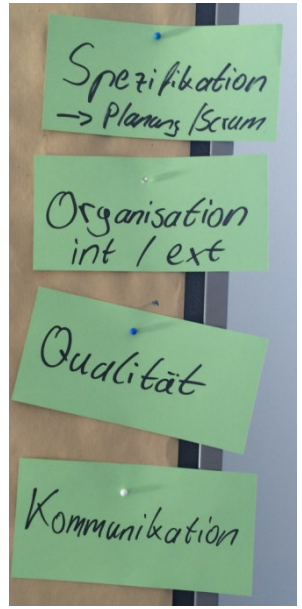
Entwickler(in) / Teammitglied

- Verantwortung
  - Entwicklung der vom Kunden gewünschten Features
  - Abstimmung der Anforderungen mit dem Kunden
    (das kann der Product Owner nicht alleine)
  - Hohe Qualität: Verständlichkeit, Qualitätskriterien
    z.B. von Uncle Bob aus Clean Code / Clean Architecture
  - Hohe Qualität: Performance, Security, Betreibbarkeit
- Tätigkeiten / Ergebnisse
  - Quelltexte, Testautomatisierung, Bearbeitung der Merge-Requests (Code-Review)
  - Modultests mit XUnit, Integrationstests
  - Explorative Tests

Klassische Rollen

- Idee: Höhere Produktivität durch Arbeitsteilung und Spezialisierung (-> Fließband)
- Ursprung: Scientific Management (F.W. Taylor)
  - Projektleiter teilt Arbeit auf (push - Prinzip)
  - Team arbeiten Aufgaben ab

Projektleiter(in)



- Verantwortung
  - ***Kunde fühlt sich gut betreut***
  - Termine werden eingehalten
  - Risiken werden aktiv verfolgt
  - ***Team kann produktiv Arbeiten***
    (Happiness-Index verfolgen, Retrospektiven durchführen)
  - Budget wird eingehalten, bzw. bewusst überschritten
- Tätigkeiten / Ergebnisse
  - Einladung ***Kundentermine*** mit Agenda
  - Versenden des ***Ergebnisprotokolls*** nach Kundentermin
  - Absprachen mit dem Kunden (erste Anlaufstelle)
  - Koordination des Teams (kooperative Aufgabenverteilung)
  - Beistellungen kontrollieren und einfordern

Technische(r) Architekt(in)

- Verantwortung
  - Team kann ***produktiv entwickeln***
  - Technische Risiken werden verfolgt
  - Software ist baubar, da die Arbeit im Team verteilt werden kann (Komponentenschnitt)
  - Software hält Qualitätskriterien ein (Performance, Security, …)
- Tätigkeiten / Ergebnisse (Sie bauen die Kathedrale!)
  - ***Architektur der Software***
    (Verteilung, Schichten, Code-Struktur, …)
  - Verwendete Frameworks, Programmiersprache
  - Technischer Durchstich
  - ***Software Entwicklungsumgebung (IDE, CI/CD, Merge-Requests, Git)***
  - Technische Reviews
  - Konzept zum Test von Qualitätsanforderungen

Tester(in)

- Verantwortung
  - **Kunde bekommt ausschließlich geprüfte und überarbeitete Dokumente und Code vom Team**
  - ***Qualität der Dokumente***
    (Formale Qualität, Lesbarkeit, Vollständigkeit, …)
  - ***Qualität der Quelltexte,*** Qualität der Software
- Tätigkeiten / Ergebnisse
  - Akzeptanzkriterien zu jeder User Story, d.h. der / die Tester(in) wirkt bei der Erstellung der User Storys mit!
  - Qualitäts- / Akzeptanzkriterien sind klar für Dokumente, Code, …
  - Mitwirkung im Konfig. Management

Hardware Architekt(in)

- Verantwortung
  - Individuell gefertigte Hardware rechtzeitig verfügbar
  - Technische Risiken werden verfolgt
  - Hardware hält Qualitätskriterien ein
- Tätigkeiten / Ergebnisse
  - ***Architektur des Gesamtsystems***
    (HW/SW-Schnitt, Code-Struktur auf Microcontroller, …)
  - Technischer Durchstich
  - ***Software Entwicklungsumgebung (IDE, Test-Umgebung, Debugger, …)***
  - Technische Reviews
  - Konzept zum Test von Qualitätsanforderungen

Usability Ingenieur(in)

- Verantwortung
  - Projektergebnis ist ***nützlich für den Kunden***
  - Projektergebnis ist ***benutzbar*** (= Gebrauchstauglich)
  - Kunde hat ein positives Nutzungserlebnis (-> UX)
  - Organisiert das ***Usability Testessen***
- Tätigkeiten / Ergebnisse
  - Persona Hypothesen, Bedürfnisse der Kunden entdecken
  - Hypothesen über den Nutzen und Nutzung
  - Verifikation der Hypothesen
  - Bedürfnisse / Verhalten der Personas / Nutzer analysieren
  - GUI-Design
  - Usability Tests speziell für das ***Usability Testessen***

IT-Betrieb (OPS)

- Verantwortung
  - Die Software / das Produkt kann später beim Kunden / in der Cloud betrieben werden
  - Klärung, was genau zum Betrieb gehört (Betriebsdaten vom Server genauso wie Nutzerfeedback z.B. mit Google Analytics)
- Tätigkeiten / Ergebnisse
  - Konzept zum Logging (was kommt in die Log-Datei?)
  - Konzept zum Monitoring des Systems (z.B. wie stellen wir fest, ob der Server noch läuft, wie finden wir heraus, wie viele Nutzer gerade eingeloggt sind)
  - IT-Sicherheit im Betrieb (Datenschutz/EU-DSVO, Ausfallsicherheit, Datensicherheit, Backup, ….)

Wichtig!

- Teams sollten heterogen besetzt sein
  Heterogen: beide Geschlechter, jung und alt, oberflächlich und Erbsenzähler, extrovertiert/introvertiert, viele Nationen, viele Religionen …
- Stärken sie ihre Stärken! (-> vgl. F. Malik)
  Alle Schwächen ausgleichen = Im Mittelmaß landen
  Setzen sie jede(n) im Team orientiert nach Stärken ein
- Arbeiten sie im Diskurs (-> vgl. Teufelsquadrat nach H. Sneed)
  z.B. Product Owner „Features“ vs. Tester „Qualität“
  z.B. Projektleiter Budget vs. Architekt „Qualität“
  z.B. Product Owner „Features“ vs. Usability Eng. „Qualität“
- Für jedes Ergebnis muss es eine(n) eindeutigen Verantwortlichen geben