ZKI HERBSTTAGUNG 2019

HOCHSCHULE DÜSSELDORF

SW-Entwicklung an der RWTH am Beispiel des QM-Systems Lehre

… und der Einfluss von Governancestrukturen

Bernd Decker

IT Center | RWTH AACHEN UNIVERSITY

Agenda

- Das Qualitätsmanagementsystem Lehre der RWTH Aachen
- Governance in IT Projekten
- Agile Softwareentwicklung
- Projektverlauf

Akkreditierung

- **Akkreditierung**
  - Bewertungsverfahren für Studiengänge zur Gewährleistung eine hohen Qualität von Lehre und Studium

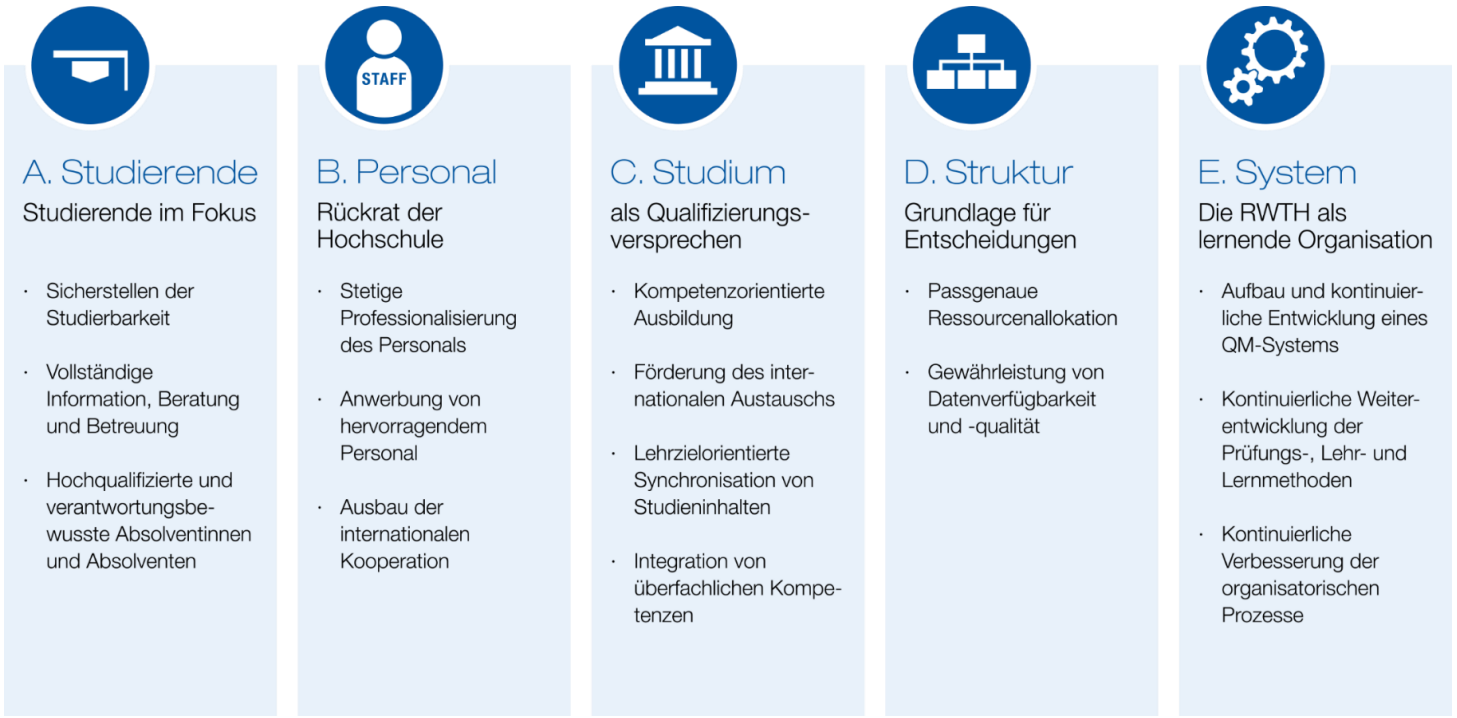
- **Programmakkreditierung** (seit 2005)
  - Die Studiengänge der RWTH werden in separaten Verfahren durch externe Akkreditierungsagenturen einzeln zertifiziert

- **Systemakkreditierung** (seit 2018)
  - Qualitätsmanagement- und Steuerungssystem für die Lehre wird als Ganzes zertifiziert
  - Alle Studiengänge der RWTH sind akkreditiert sofern sie das QM-System Lehre durchlaufen
  - Die RWTH mit ihren eigenen Verfahren und Instrumenten der Qualitätssicherung akkreditieren

- Voraussetzung für erfolgreiche Systemakkreditierung ist ein Steuerungs- und Qualitätsmanagementsystem, das die Qualitätsstandards der Studiengänge gewährleistet

Grundlage für das QM-System - Qualitätsziele auf der Grundlage von Kriterien, Kennzahlen und Indikatoren

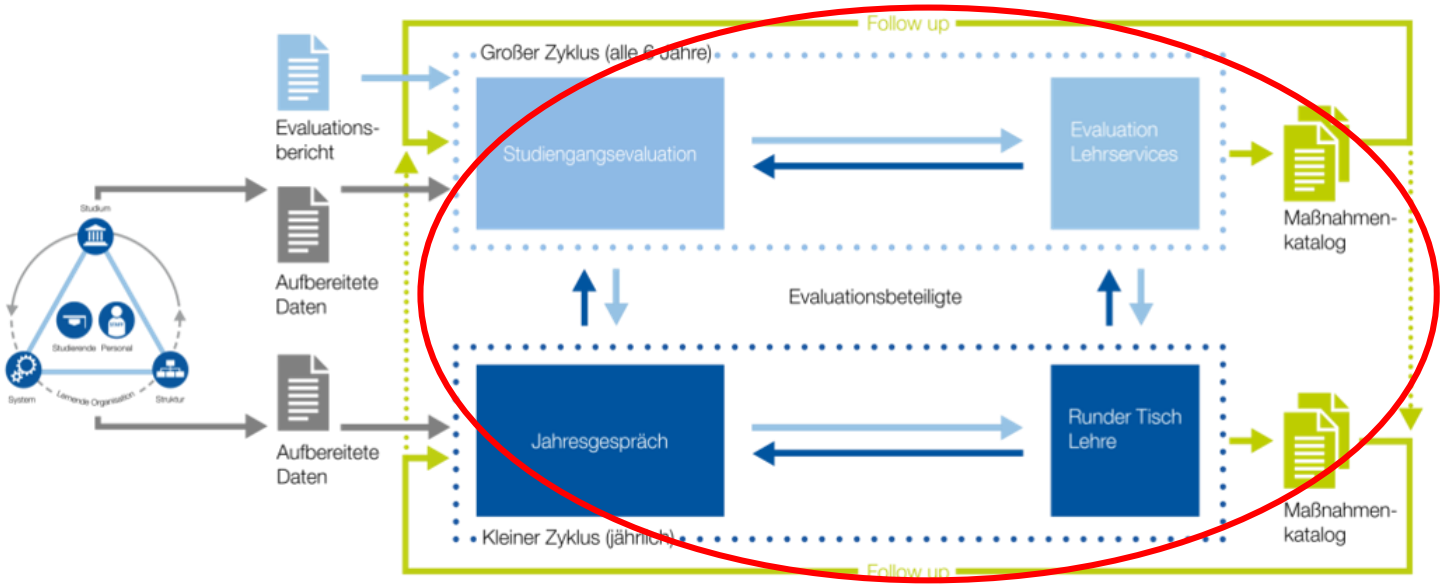
- 5 Qualitätszielbereiche
- 3 Qualitätsziele je Qualitäts-zielbereich
- Jedes Qualitätsziel wird durch 3 - 5 Kriterien näher beschrieben
- Datenmaterial in Form von Kennzahlen und/oder Indikatoren für jedes Kriterium
- Insgesamt über 180 Kennzahlen und Indikatoren



Etablierung eines RWTH-weiten gemeinsamen Verständnis was gute Lehre auszeichnet

Qualitätszyklen zur Steuerung (Plan, Do, Check, Act)

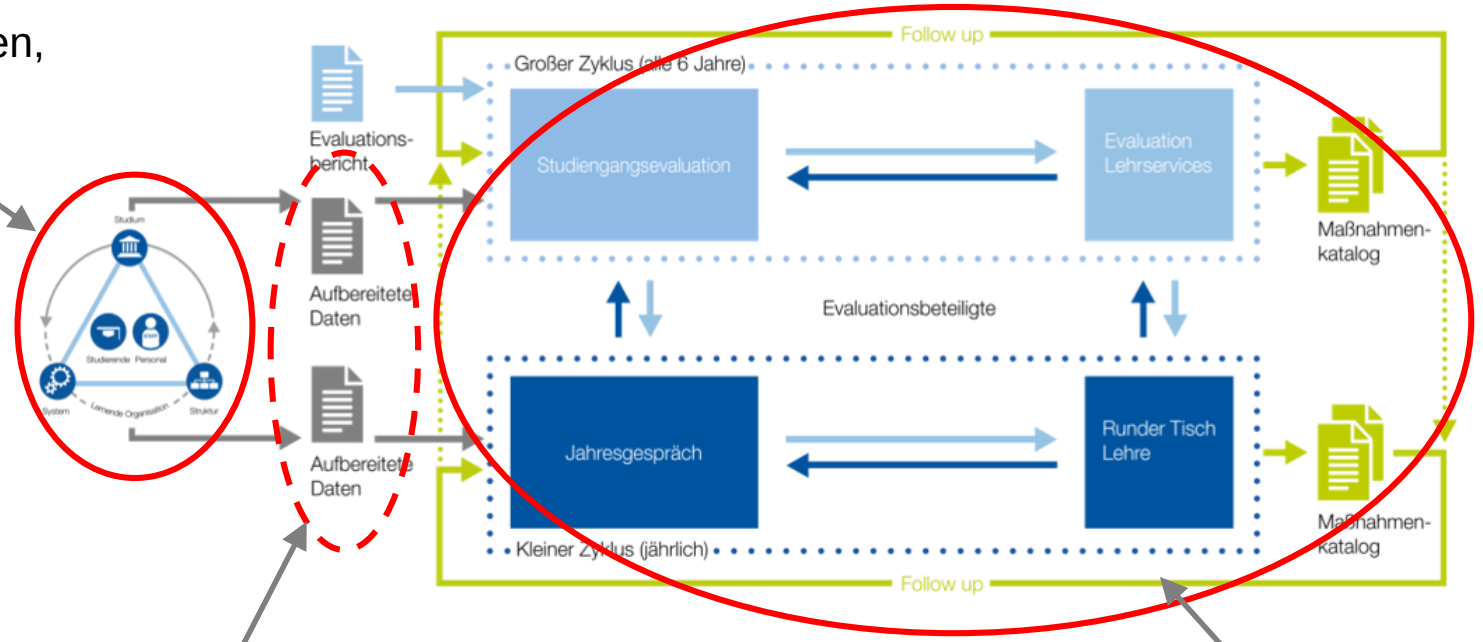
- **Studiengangsevaluation** (alle 6 Jahre) Detaillierte Analyse der Studiengänge und Festlegung fachspezifischer Zielwerte, mit Evaluationsbericht, Maßnahmenkatalog und Audit mit externer Begutachtung
- **Jahresgespräch** (jährlich) Gespräche zw. Prorektor Lehre und Studierenden sowie Prorektor Lehre und Fakultäten mit Festlegung und Überprüfung der Maßnahmen



- **Evaluation Lehrservices** (alle 6 Jahre) Standardisierte Befragung der Lehrenden, Studierenden und Serviceeinheiten Lehre mit anschließendem internen Review zur Qualität der Services im Bereich Lehre, Festlegung von Maßnahmen
- **Runder Tisch Lehre** (jährlich) Gespräche mit den Serviceeinheiten Lehre, Prorektor Lehre, Kanzlervertreter, Studierenden und akademischer Seite mit Festlegung und Überprüfung der Maßnahmen

Qualitätsmanagementsystem Lehre

Qualitätsziele auf der Grundlage von Kriterien, Kennzahlen und Indikatoren

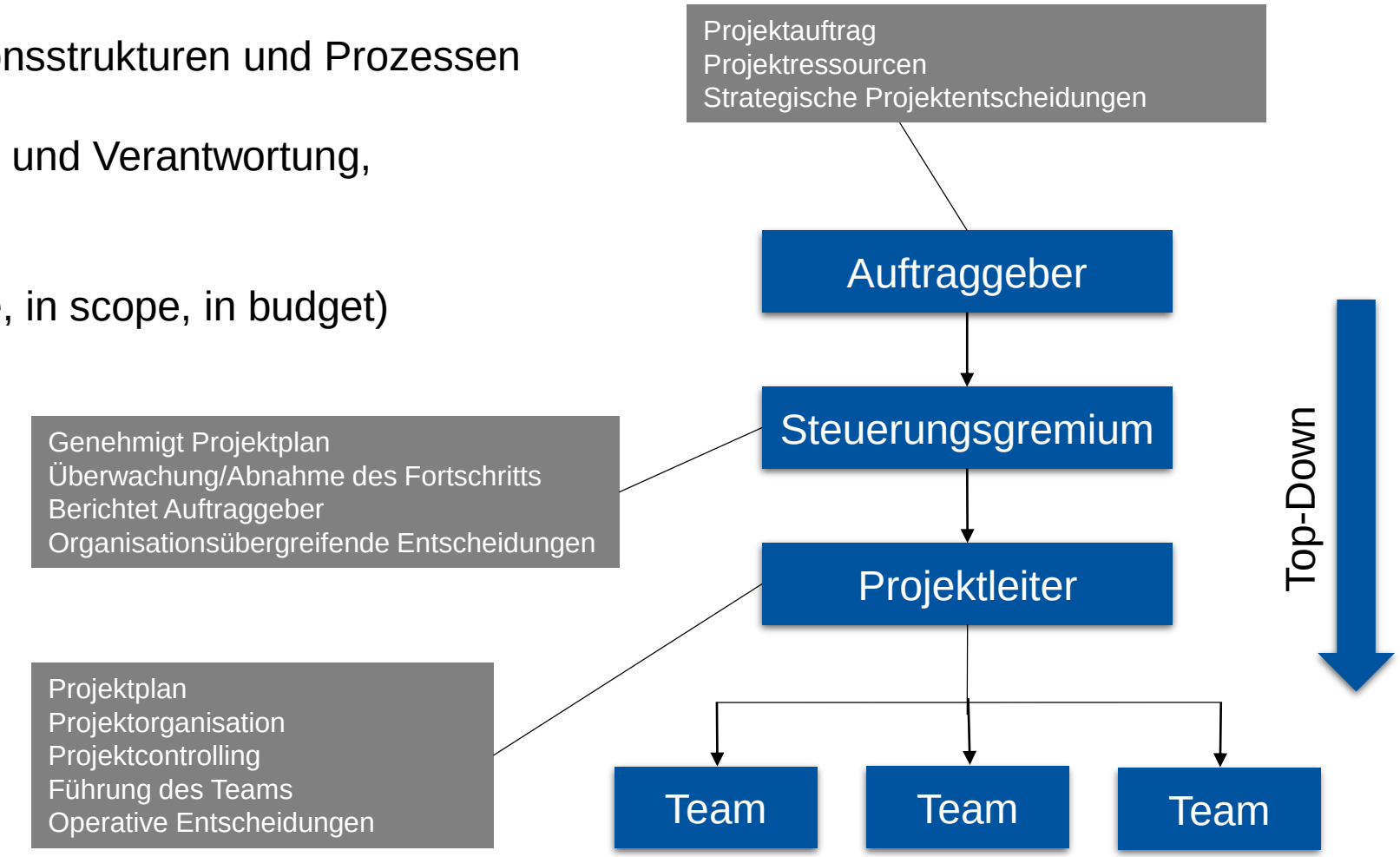


Datencockpit

Qualitätszyklen zur Steuerung

Projekt-Governance

- Besteht aus Führung, Organisationsstrukturen und Prozessen
- Gibt Richtung vor, definiert Rollen und Verantwortung, Evaluiert, Überwacht
- Ziel: Erfolgreiches Projekt (in time, in scope, in budget)
- Risiken:
  - Budgetüberschreitung
  - Zeitplanüberschreitung
  - Fehlerhafte Software
  - Das Falsche liefern



Agile Softwareentwicklung - Leitbild

Manifest für Agile Softwareentwicklung (https://agilemanifesto.org/)

- **Individuen und Interaktionen** wichtiger als Prozesse und Tools
- **Laufende Software** wichtiger als umfassende Dokumentation
- **Zusammenarbeit mit dem Kunden** wichtiger als Vertragsverhandlung
- **Reagieren auf Veränderung** wichtiger als das Befolgen des Plans

Das Ziel in einem agilen Projekt besteht darin, die bestmögliche Software unter Berücksichtigung der Kosten, der Funktionalität, der Zeit und der Qualität abzuliefern

Agile Development - Scrum

Bildquelle: https://www.it-agile.de/fileadmin/PDF/Scrum_Intro_V1.2.pdf

Product Owner: Entwickelt Produkt-Vision mit Stakeholdern und Anwendern, repräsentiert den Kunden, priorisiert Pordukt-Features, verantwortet den Produkterfolg, nur eine Person

Scrum Master: Hilft Product Owner und Entwicklungsteam Scrum effektiv anzuwenden, identifiziert Hindernisse und beseitigt sie, moderiert Meetings, fördert Eigenverantwortung

Entwickler: Zuständig für die eigentliche Entwicklung, cross functional, selbstorganisierend, tragen Verantwortung für das Inkrement

Stakeholder: *jeder, der Interesse am Produkt oder Einfluss auf die Entwicklung hat: Kunden, Anwender, Manager, Peronalrat, etc.*

Abkehr von der „command-and-control“-Organisation hin zu selbstorganisierten Teams

Agile Development

- Softwareentwicklung ist ein kreativer Prozess, sowohl was das Finden von Anforderungen betrifft als auch deren Umsetzung in lauffähige Software

- Komplexe Softwareprojekte sind immer Unsicherheiten behaftet und nicht vollständig vorherzusehen
  - Langfristige, detaillierte Pläne suggerieren lediglich ein Gefühl von Sicherheit

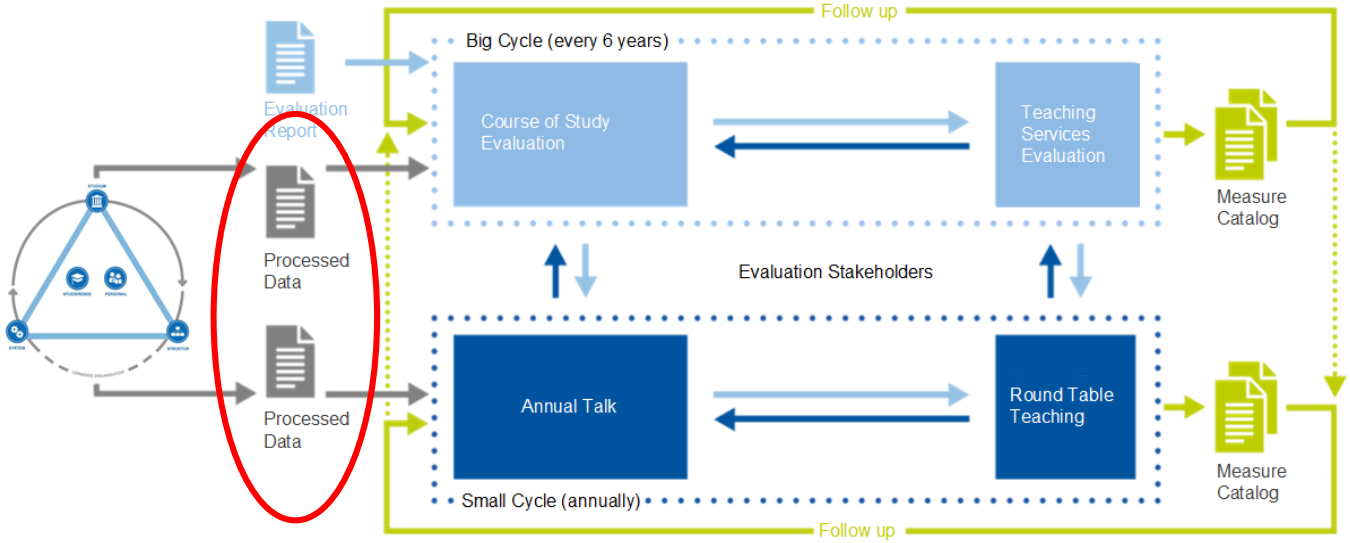
- Rückkopplung mit Kunden und Flexibilität ist entscheidend, um die Unsicherheit zu reduzieren
  - Kunde muss Zeit investieren um regelmäßig und schnell Feedback zu geben

- Beschäftigung ist kein Indikator für den Fortschritt, sondern Ergebnisse
  - Transparenz durch Burndown Chart, Sprint Backlog und Sprint Review, kein weiteres „Management Reporting“

QM-System Lehre – Softwareunterstützung

Datencockpit:

- Systematische Datengrundlage für das Qualitätsmanagement
- Komfortabler Zugriff auf alle notwendigen Daten zur Bearbeitung und Bewertung für die Evaluationsbeteiligten
- Webbasierte parametrisierbare Reports auf Basis von Kennzahlensteckbriefen
- Daten aus dem gesamten Bereich Lehre
- Rollen & Rechte basiert



Projektstart – Vorarbeit (Q1 2016)

- Festlegen des Entwicklerteams
  - Kompetenzen: Entwicklung, UX und Tests
- Bestimmung Product Owner und Scrum Master
- Backlog befüllen
  - User Stories und Kennzahlensteckbriefe als Grundlage für funktionale Anforderungen
  - Milestones anhand der Rahmenbedingungen durch Systemakkreditierung
  - Minimal Marktfähiges Release (must haves)
- Infrastruktur klären
  - SharePoint mit integrierten SQL Server Reporting Services
- Kommunikation mit Auftraggeber/Fachabteilung
  - Regelmäßiges Meeting entsprechend des Sprintzyklus
  - Definierte Ansprechpartner
  - Anforderungen über ProductOwner
  - Iterative Entwicklung, Steuerung über Sprint Review

Projektverlauf - Softwareauslieferung

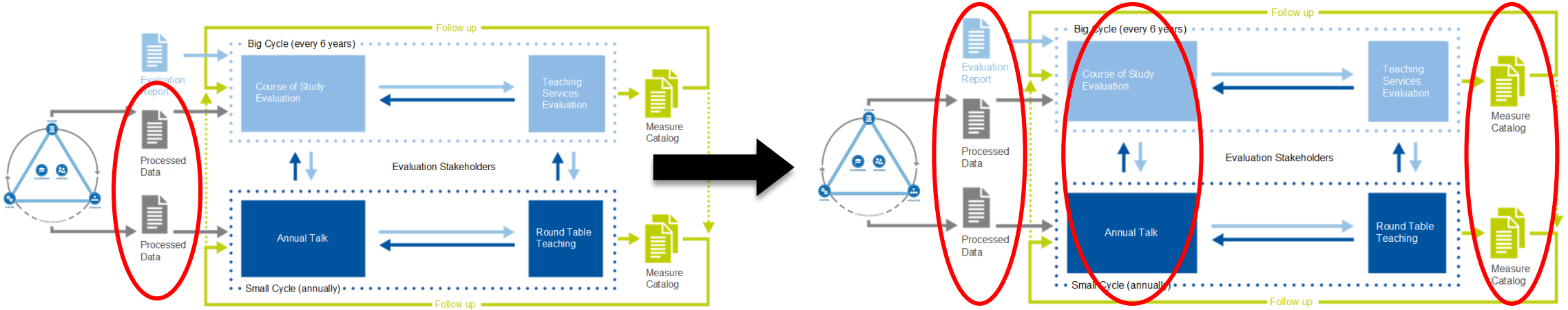
- Problem:
  - Fehlende Kompetenzen im Entwicklerteam im Bereich IT Betrieb / Operations
  - Übergabe zwischen den Abteilungen notwendig
  - Unterschiedliche Prozesse und Zielrichtungen
  - Zeitaufendig, Product increment nicht zum Sprint Review verfügbar
  - Rückkopplung fehlt, bzw. Leerlauf bei den Entwicklern
  - Fehlende Steuerung, Unsicherheit

- Lösung: Continuous delivery
  - Standardisierte automatisierte Build, Deployment und Test Prozesse (bis Pre-Production Stage)
  - Automatisierung ermöglicht schnelle, zuverlässige und wiederholbare Deployments
  - Kurze Deploymentzyklen

- Folgen: Verschiebung von klassischen Ops-Tätigkeiten (und Verantwortung) zu Dev
  - Auslieferung in Sprint-Zyklen

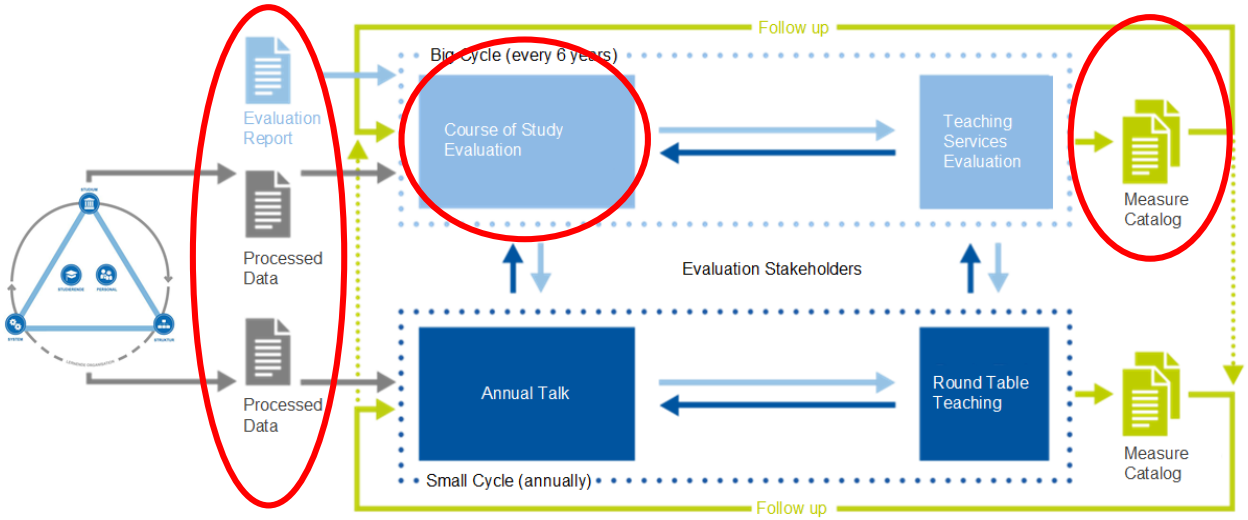
Projektverlauf – Veränderte Anforderungen

- Zusätzliche funktionale Anforderungen - Abbildung des Prozesses „Ziele setzten und Nachverfolgen“ mit integrierten Kennzahlen und Indikatoren
  - Studiengangsevaluation
  - Jahresgespräche
  - Maßnahmenkatalog
  - Evaluationsbericht



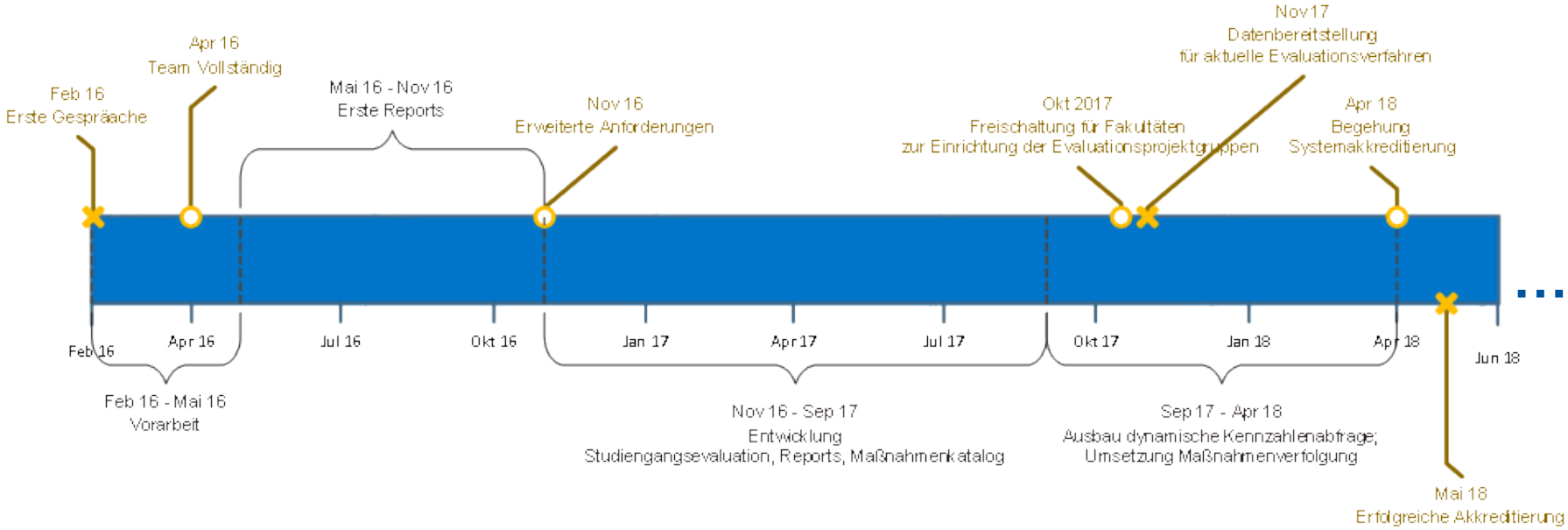
Anpassung der Features

- Gemeinsame Priorisierung
  - Anforderungen aufteilen
  - Was sind notwendig Anforderungen für die Akkreditierung, was sind Komfort Anforderungen?
  - Wo gibt es ggf. manuelle Workarounds als Plan B?
- Reduktion auf:
  - Studiengangsevaluation
  - Maßnahmenkatalog
  - Evaluationsreport
  - Bereitgestellte Datengrundlage teilweise nicht automatisiert sondern manuell hinterlegt



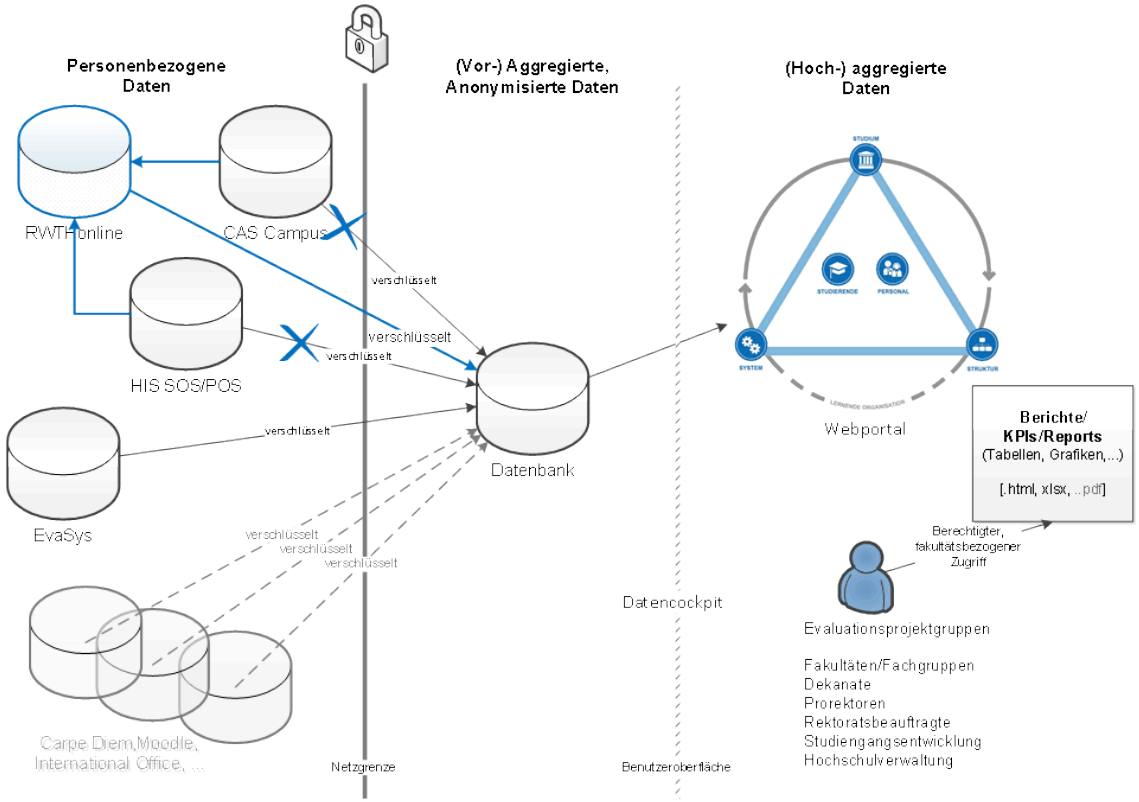
Projektverlauf - Timeline

- Projektstart / Vorarbeit: Q1 2016
- Start Entwicklung Datencockpit: Q2 2016
- Erweiterung Team, neuer Product Owner: Q4 2016
- Pilot Gesamtprojekt Systemakkreditierung (Go Live Datencockpit): WS 2017/2018
- Begehung Systemakkreditierung: 04/2018
- Erfolgreiche Akkreditierung: 05/2018



Projektverlauf – Einführung neues Campusmanagement System

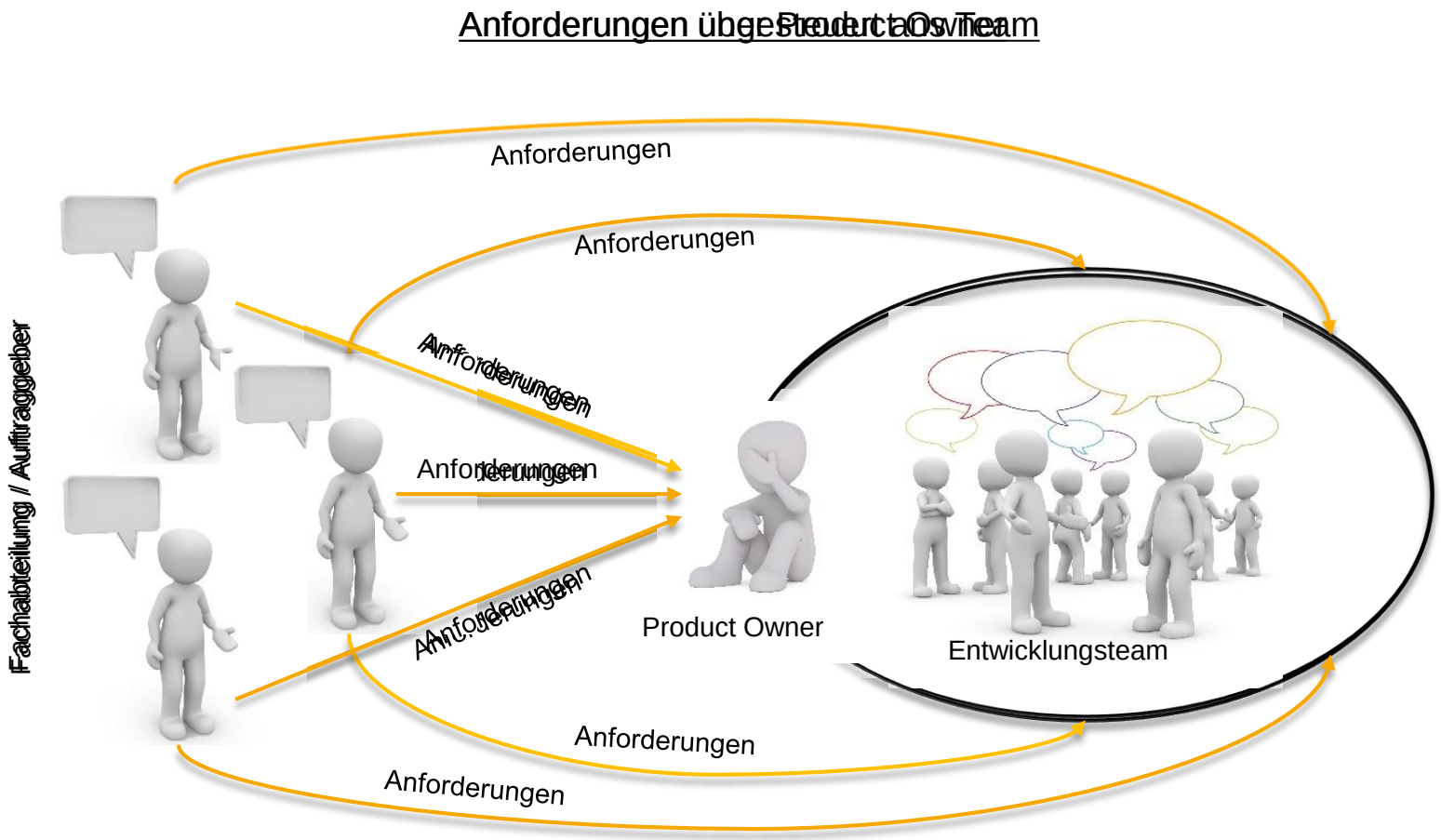
- Einführung RWTHonline WS18/19
  - Änderung der primären Datenquellen für das QM-System Lehre
  - Veränderte Daten, Schnittstellen und Technologien
  - Abstraktionsschicht um Anpassungen in der Anwendung zu verhindern



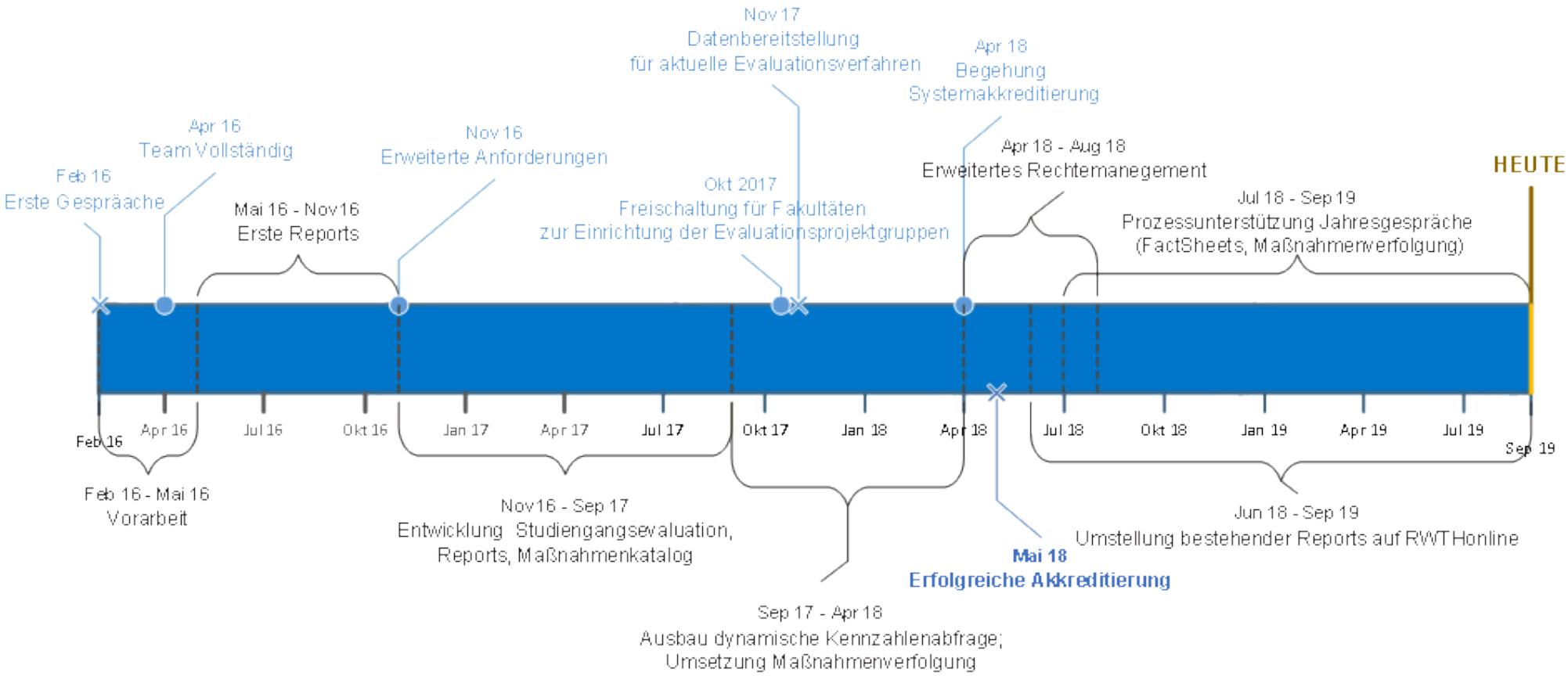
Projektverlauf – Einführung RWTHonline

- Umstellung
  - Fehlendes Wissen zu Daten und Datenmodellen
  - Experten aufgrund der Einführung ausgelastet
  - Gleiche Modellierung über Abstraktionsschicht nicht 100% möglich
  - Daten in einigen Teilen nicht vergleichbar mit Daten aus Altsystem
  - Fehler in der Interpretation und Darstellung von Daten in umgestellten Reports
  - Entwickler kümmern sich vor allem um „Bugfixing“
  - Keine Weiterentwicklung/Vorankommen erkennbar für Fachabteilung/Auftraggeber

Projektverlauf - Kommunikation



Aktuelle Stand



Lessons Learned

- Agile SW-Entwicklung erhöht Flexibilität und Transparenz im Projekt
  - Fehlentwicklungen können frühzeitig identifiziert und beendet werden, schnelle Neuausrichtung möglich

- Aber hat Auswirkungen auf Governance Strukturen

  - Konfliktpotential zwischen klassischem Projektmanagement mit statischer Planung und Prozessdenken
    - Projektmanagement muss sich darauf einlassen / anpassen

  - Hohe Einbeziehung/Mitwirkung des Auftraggebers während des gesamten Projekts
    - Beteiligung an inkrementeller Planung/Priorisierung
    - Regelmäßige Überprüfung der Sprint Ergebnisse (Rückkopplung)

  - Governance Strukturen außerhalb des Projekts
    - Zusammenspiel mit anderen Frameworks wie ITIL

  - Selbstorganisierte Teams
    - Mehr Selbstverantwortung für einzelne Mitarbeiter benötigt eine Umgewöhnung
    - Fehlerkultur / Umgang mit Fehlern ein wesentlicher Faktor

Vielen Dank
für Ihre Aufmerksamkeit

Fragen / Diskussion

Bernd Decker
decker.@itc.rwth-aachen.de