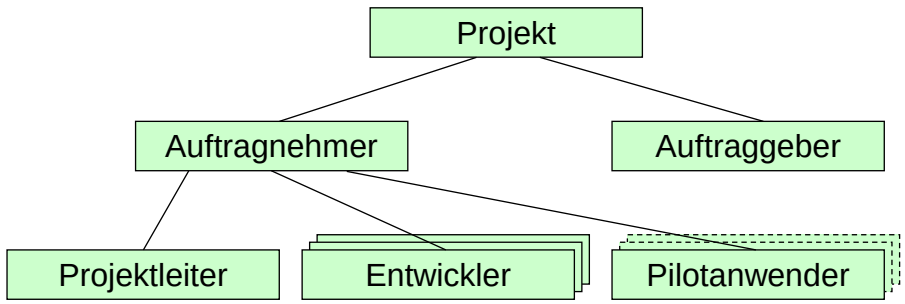
Moderne Softwareentwicklungs-Prozesse: Rollen, Dokumente, Werkzeuge

Herwig Mayr

Fakultät für Informatik/Kommunikation/Medien
FH OÖ Campus Hagenberg

Moderne Softwareentwicklung – Rollen



Projektleiter / Pilotanwender: Positionierung

Projektleiter (PL)	Pilotanwender (PA)
„organisatorischer Projektleiter“	„inhaltlicher Projektleiter“
prozessbezogener Projektleiter	produktbezogener Projektleiter
auch *Scrum Master*, *Gewissen*, *Coach*, *Big Boss*	auch *Product Owner*, *On-site Customer*, *Virtual Customer*, *Tester*
1 mit Stv.	**1 – n** mit Stv. (pro Anforderung genau 1!)
teilzeitlich! passt gut zu RM, schlecht zu PR	**teil-** oder **vollzeitlich**! passt gut zu PR, schlecht zu RM

Aufgabenverteilung PL / PA / Entwickler

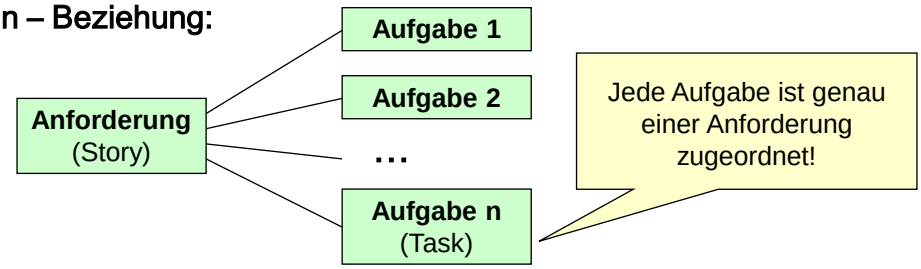
Projektleiter: überwacht Ablauf,
versteht, steuert und dokumentiert Prozess,
teilt Aufgaben zu

Pilotanwender: bewahrt Produktsicht (≠ *Entwickler* !)
treibt dadurch Entwicklung voran
(Anforderungsauswahl!)
erstellt Testfälle

Entwickler (<= 7): schätzen und wählen Aufgaben
entwickeln (primär D-I-T)
übernehmen andere Aufgaben (z.B. System-admin., Weiterbildungskoord.) teilzeitlich
(*Gruppenorganisation ist vorgehensspezifisch!*)

Anforderungen und Aufgaben: Zuordnung

1:n – Beziehung:



Beispiel:

Anforderung: Projektverlaufsdarstellung
(„Ich möchte den Projektverlauf grafisch dargestellt haben.“)

Aufgaben: Aufwandslinie zeichnen,
Filter über Aufgaben erstellen,
Formeln für Auswertekurven festlegen,
…

Anforderungen und Aufgaben: Vergleich

Kriterium	**Anforderung**	**Aufgabe**
Phase	Analyse (WAS?)	Design (WIE?)
Festlegung	AG	AN
Priorisierung	AG, PA	AN (PL, PA)
Abschätzung	PA [Arbeitstage]	Entw. (1 | Gruppe) [AEH]
Verantwortlicher	PA (1)	Entwickler (>= 1)

Transition von Dokumentationsmodellen (I)

Effizienter Werkzeugeinsatz ist bei agilen Methoden essenziell!

Wichtiger bei traditionellem Vorgehen	Gleich wichtig	Wichtiger bei agilem Vorgehen
	Organigramm, Stellenbeschreibungen	
	Projektbibliothek, Projekttagebuch, Protokolle	
Fortschrittsberichte		Zwischenprodukte, Prototypen (Protokolle)
	Konfigurationsmanagement (Werkzeug!)	
	Qualitätsmerkmale	
	Zielbeschreibung	
Lastenheft		Auftrag, Anforderungskatalog (Werkzeug!)
Gesamtplan		Grobplan, kurzfristiger Feinplan (Werkzeug!)
		Risiko-Vorsorgeplan

Transition von Dokumentationsmodellen (II)

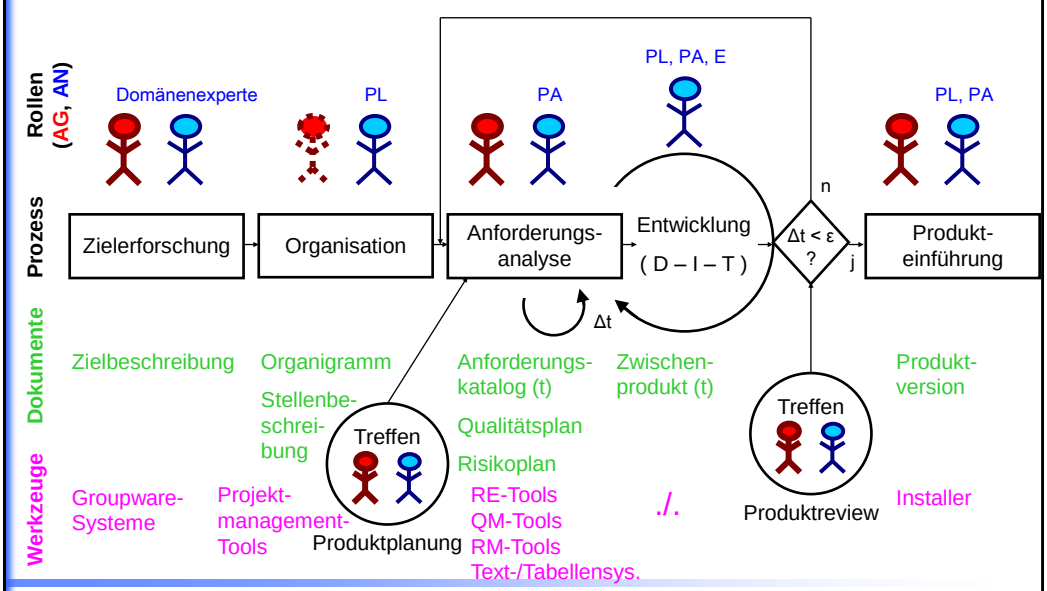
Wichtiger bei traditionellem Vorgehen	Gleich wichtig	Wichtiger bei agilem Vorgehen
Benutzerdokumentation		Zwischenprodukte, Prototypen
Pflichtenheft		Designmodell (Werkzeug!), durchg. Modellsprache
	Quellcode	
Systemdokumentation		(Anwendungs-)Daten
	Lauffähiges Programm	
		Testplan, Testsuite
Fehlerliste, Fehlerbericht		Fehlerbehandlung (Werkzeug!)
	Fehlerlogbuch	
	Betriebsversion	
	Installations-, Inbetriebnahmeprotokoll	
	Abnahmeprotokoll	
Abschlussbericht		Wartungs-, Pflegeplan
	Projektarchiv	

Dokumente und Werkzeuge

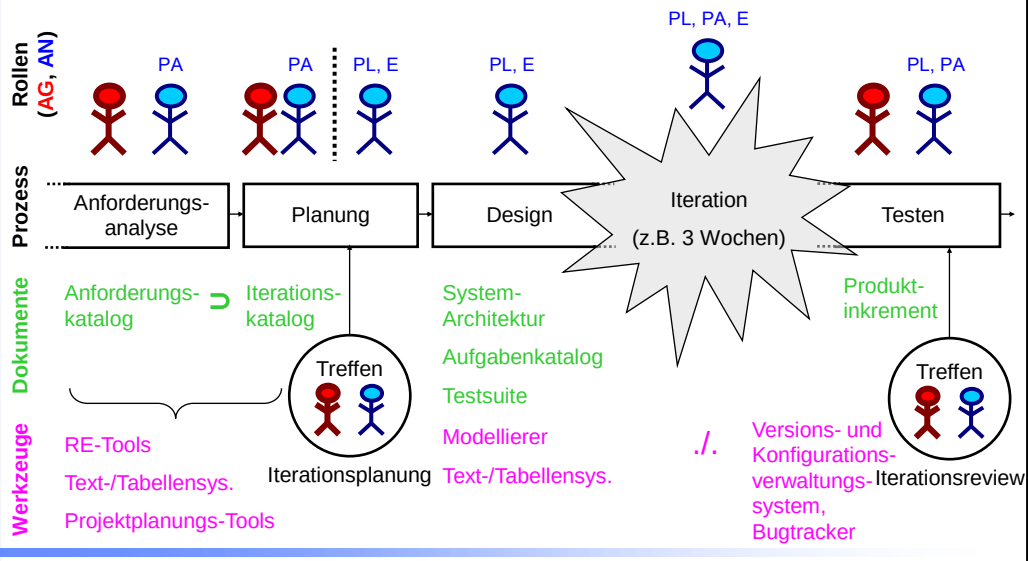
Dokument	Sichtbarkeit	Werkzeug
Projektbibliothek	I	CM-Tool, QM-Tool / Tab.
Organigramm	I	Textv.
Stellenbeschreibungen	I	Textv.
Agenden, Protokolle	I / E	Textv.
Prototypen	I / E	div. Prot.-Tools, Entw.umg.
Risikoplan	I	RM-Tool / Tab.
Zielbeschreibung	E	Textv.
Tagebuch	I	Textv./Tab./DB
Anforderungen	E	Textv./Tab./DB
Projektplan	I	PM-Tool
Aufgaben	I / E	Textv./Tab./DB
Fortschrittsdokumentation	I / E	Textv.
Benutzerdokumentation	E	Textv.
Quellcode	I / E	Entw.umg.
Systemdokumentation	I / E	Textv.
Testdokumentation	I / E	Test-Tool, Bugtracker
Programm	E	Entw.umg.
Abschlussbericht	I	Textv.
Archiv	I	-

Alle „**E**“ sind eigene (externe) *Anforderungen*, die zu *Aufgaben* führen!

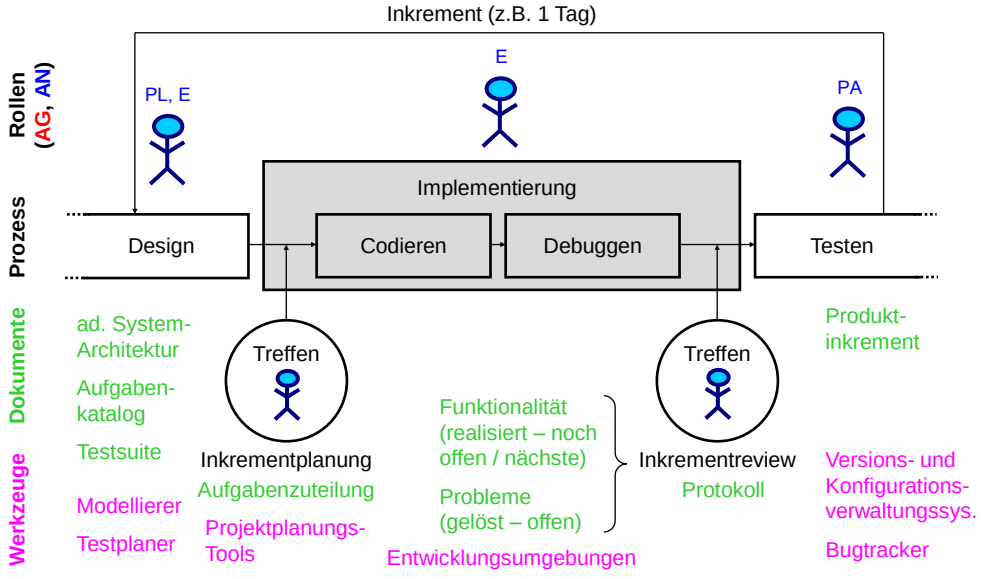
Werkzeugunterstützung im PE-Zyklus (I)



Werkzeugunterstützung im PE-Zyklus (II)



Werkzeugunterstützung im PE-Zyklus (III)

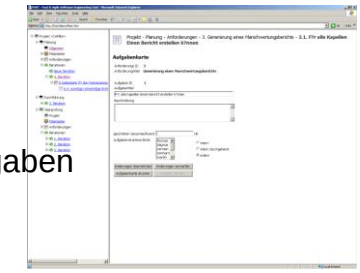


Moderne Werkzeuge – Struktur

Projekt:

Planung

- Allgemein
- Mitarbeiter
- Anforderungen – Aufgaben
- Iterationen



Durchführung

- Iteration – Anforderungen – Aufgaben

Überprüfung

- Projekt
- Mitarbeiter
- Anforderungen – Aufgaben
- Iterationen – Anforderungen – Aufgaben

Planung “Projekt”

Projekt – Planung – Allgemein:

- Projektname
- Arbeitstage/Iteration
- Arbeitseinheiten/Tag
- Bezeichner für Arbeitseinheiten

Planung “Anforderungen”

Projekt – Planung – Anforderungen:

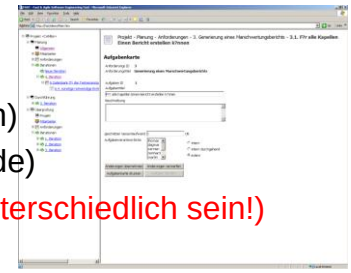
- ID (wird fortlaufend vergeben)
- Erstellungsdatum
- Priorität
- Titel
- Beschreibung
- geschätzter Gesamtaufwand (Tage)
- verantwortlicher Pilotanwender

Planung “Aufgaben”

Projekt – Planung – Anforderungen – Aufgaben:

- ID (wird fortlaufend vergeben)
- Titel
- Beschreibung
- geschätzter Gesamtaufwand (Arbeitseinheiten)
- Aufgabenverantwortliche
- Typ (intern/intern durchgehend/extern)

Planung “Iterationen”

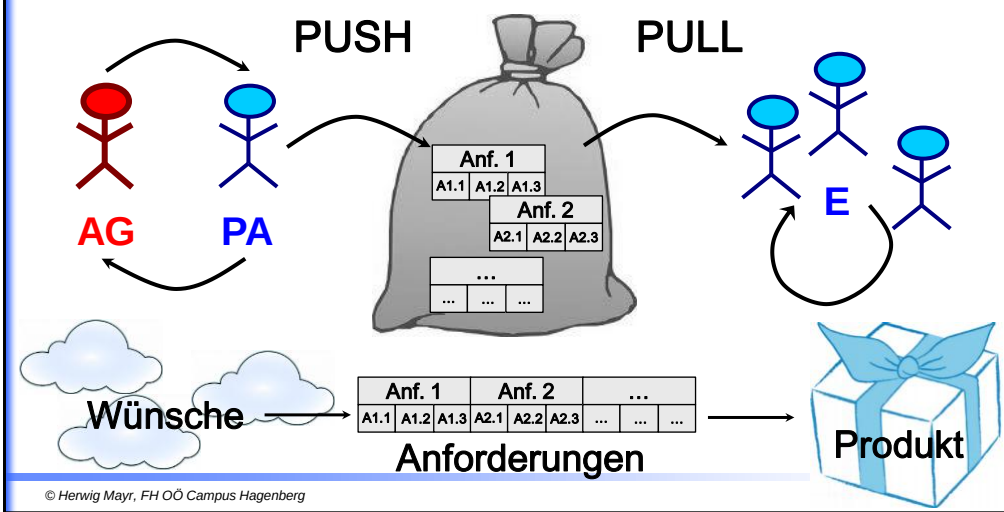


Projekt – Planung – Iterationen:

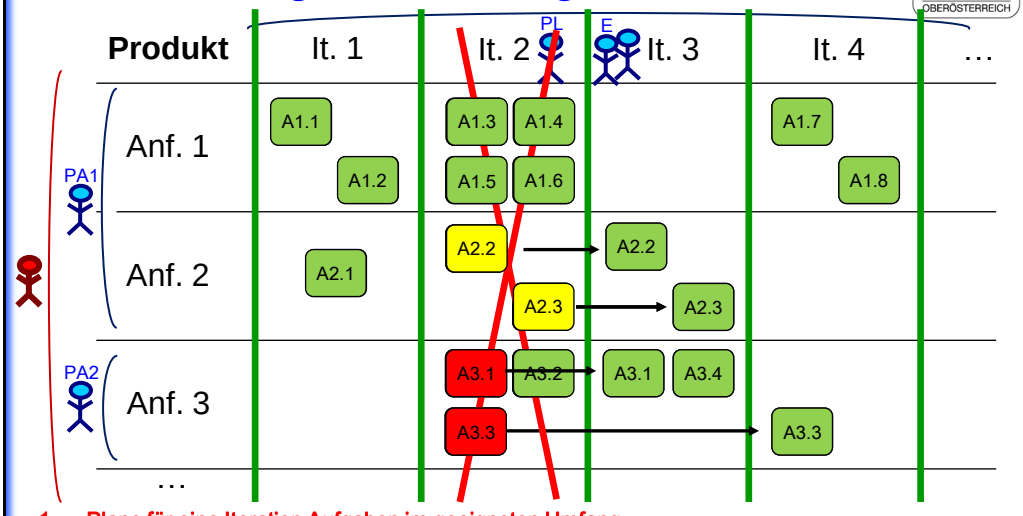
- Nr. (wird fortlaufend vergeben)
- Arbeitstage (inkl. Beginn, Ende)
  - (können je Iteration unterschiedlich sein!)
- Leistung/Aufwand/Overhead
- zugeordnete Aufgaben
  - (NICHT Anforderungen!)

Planung: Kombiniertes Push-Pull-Prinzip

Pufferfunktion des Anforderungs- bzw. Iterationskatalogs



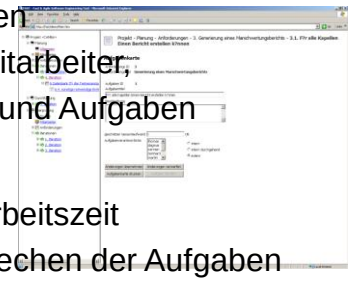
Durchführung: Anforderungen vs. Iterationen



1. Plane für eine Iteration Aufgaben im geeigneten Umfang.
2. Schließe zumindest die Aufgaben einer Anforderung je Iteration ab.
3. Schließe nicht abgeschlossene Aufgaben in der nächsten Iteration ab.
4. Entscheide für jede abgebrochene Aufgabe, ob und in welcher Iteration sie fortgeführt wird.

Durchführung: Workflow

Planung: beliebig viele Iterationen
auf Basis definierter Mitarbeiter,
Anforderungen und Aufgaben

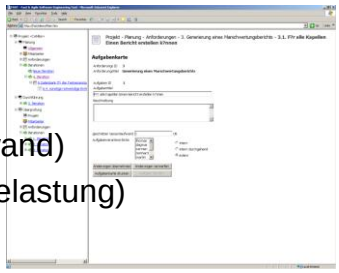


Durchführung: genau **EINE** Iteration
durch Eintragen der Arbeitszeit
und Abschließen/Abbrechen der Aufgaben
(Achtung auf obsolete Aufgaben!)

Überprüfung: aller abgeschlossenen Iterationen
sowie der aktuellen Iteration
(Ampelmetapher)

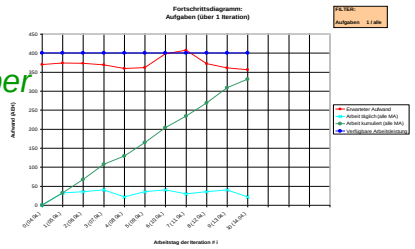
Überprüfung

Laufende Überprüfung von:



- Fortschritt vs. Zielerreichung (Arbeitszeit vs. geschätzter Restaufwand)
- Aufwandsverteilung (gleichmäßige Belastung)
- Schätzfehler
- Management-Overhead
- …

→ *mit Kennzahlen, besser aber mit Grafiken!*



Empfehlungen für einen effizienten Werkzeugeinsatz

1. Trenne den **Entwicklungsprozess** von Werkzeugen, Methoden und Sprachen!
2. Wähle durchgehend nutzbare, integrierbare **Werkzeuge**!
3. Verwende Werkzeuge vom **Projektbeginn** an!
4. Wähle Werkzeuge, die ein **iterativ-inkrementelles Vorgehen** unterstützen!
5. Berücksichtige Zeit für Werkzeug-**Schulungen**!
6. Sei vorsichtig beim **Wechsel** von Werkzeugen bzw. Releases!
7. Miss **Fortschritt** und **Zielerreichung**!

Dokumente & Werkzeuge in agilen Prozessmodellen

Erkenntnisse:

- Auch agile Methoden erzeugen eine **Anzahl von Dokumenten**.
- Dokumente müssen **kompakt** und mit **wenig Aufwand** zu erstellen sein.
- Agile Methoden benötigen häufig **mehr Werkzeugunterstützung** als traditionelle Methoden.
- Werkzeuge müssen **einfach erlernbar** und **rasch handhabbar** sein.

Ein gutes Werkzeug soll den Prozess unterstützen, nicht vorgeben!

Vorteile für die Ausbildung

Prozess:

- Software wird **iterativ** und **inkrementell entwickelt**
- Tägliche **Arbeit** wird **dokumentiert**; (kleine) **Pläne** werden häufig **adaptiert.**
- **Fortschritt** und **Zielerreichungsgrad** werden **gemessen.**

Persönlichkeitsentwicklung:

- Werkzeuge und Dokumente lehren Studierende **Disziplin.**
- Studierende entwickeln rasch ein gutes **Schätzvermögen.**

Schlussbemerkungen

- Bücher über agile Vorgehensmethoden skeptisch lesen, aber lesen!
- Vieles ist alter Wein in neuen Schläuchen (Geschäftstüchtigkeit).
- „Silver Bullet“ ist auch mit APM nicht gefunden.
- Dafür gibt es viel Freiraum für eigene Ideen (z.B. Gruppenzuteilung, variable Zykluslänge).