C1 GROUP

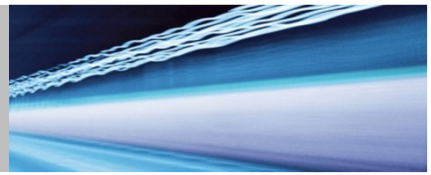
"Agil" ist inzwischen als risikoarme und zielorientierte Vorgehensweise akzeptiert. Wir begegnen kaum noch Unternehmen, die nicht zumindest ein agiles Pilotprojekt durchführen. Doch ein erfolgreiches Pilotprojekt bedeutet noch keine agile Unternehmenskultur!

Je größer ein Unternehmen, desto größer die Vielfalt der Projekte. Agil lässt sich nicht "von oben" verordnen, denn unterschiedliche Ansätze sind gerechtfertigt. Agil ist nicht gleich agil und auch nicht in allen Fällen nützlich. Wichtig ist hingegen eine unternehmensweite inkrementelle Denkweise.

Im Vortrag berichten wir auf Basis unserer umfangreichen Erfahrungen über die kritischen Punkte bei der Einführung agiler Konzepte und geben in unseren "Lessons Learned" einen Einblick in bewährte Strategien und Vorgehensweisen, sowie Beispiele für deren erfolgreiche Anwendung.

Agile Konzepte im Unternehmen verankern

Prof. Dr. Heinz Züllighoven
Dipl.-Inform. Jörn Koch
Universität Hamburg
C1 WPS GmbH



Architekturbüro

- Entwurf komplexer Architektur
- Architektur-/Qualitätsanalyse
- Architekturberatung
- Architektur-Refactoring
- Architekturmanagement

Transformationsbüro

- Prozess- & Landschaftsmodelle
- Anwendungslandschaften
- Transformationsstrategien
- Projektbüro und Koordination
- IT-Checkup

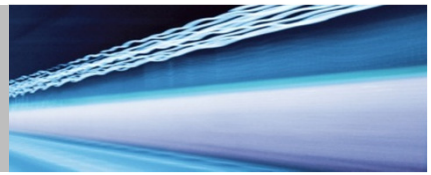
Bauaufsicht

Dienstleistung
Kompetenz
Qualität

Ingenieurbüro

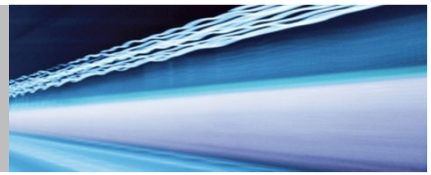
- SE-Methoden: Agile / inkrementell, SCRUM, XP, Kanban
- QS, QK, Testmanagement, Usability, Entwurfsmuster
- Technologien, Sprachen, Entwicklungsumgebungen
- Schulungen, Workshops, Coaching

Vom Pilotprojekt zur Unternehmenskultur



- Das erste agile **Pilotprojekt war erfolgreich.**
- Projektleitung und Management sind **halbwegs zufrieden.**
- Ist das schon die Grundlage für eine **neue Unternehmenskultur**?

Pilotprojekt ≠ Unternehmenskultur

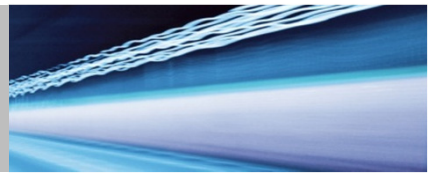


- Ein erfolgreiches agiles Pilotprojekt bedeutet noch keine nachhaltige Veränderung der Unternehmenskultur.

  + Ein **agiles Projekt ist machbar (PoC)**.
  + Eine **Entscheidung für Agilität**.
  + **Personen mit Erfahrung** in Agilität.

  – Projekt lief unter **individuellen, ggf. nicht skalierbaren Randbedingungen**.
  – Projekt ggf. nur **scheinbar** / **unzureichend agil**.
  – Projekt im Wesentlichen von **Externen** durchgeführt.

Große Unternehmen



- **Je größer ein Unternehmen, desto größer die Vielfalt:**

Technische / fachliche Projekte

Unterschiedlich erfahrene und qualifizierte Mitarbeiter

Externe / interne Mitarbeiter

Verteilte / internationale Teams

Große / kleine Projekte

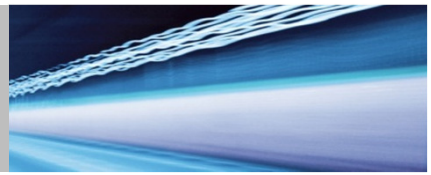
Host, Job-Verarbeitung, Desktop-Anwendungen, klassische Web-Anwendungen, moderne RIAs, Anwendungen für mobile Geräte, …

Spezialisten / Generalisten

Einzelanwendungen, komplexe Systemlandschaften

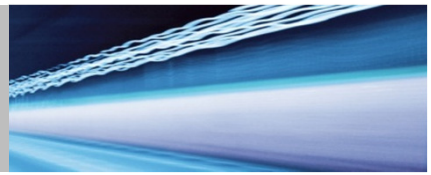
Große / kleine Teams

Keine agile Methode “nach Lehrbuch“



- **Agile Methode an Situation anpassen!**
- **Funktionierendes erhalten!**

agil ≠ agil

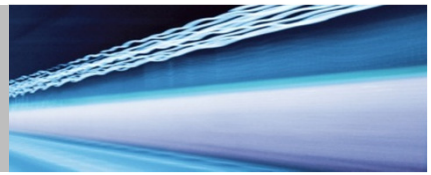


- **Agil ist nicht gleich agil!**

  - **Projektspezifische Ansätze** können gerechtfertigt oder gar notwendig sein.

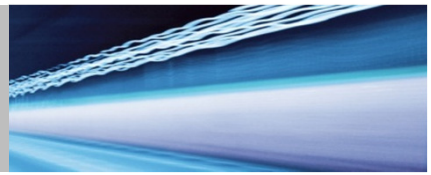
  - Aber auch der **Blick fürs Ganze** ist notwendig. Achtung bei **„pseudo-agilen“ Projekten**!

Keine agile Methode als Kochrezept



- „**Lokalisierung**“ notwendig.
  - Anforderungskanäle, Anforderungstypen, Rollen, Orga-Strukturen, Team-Kulturen, Technologien, ...
- Methodenwissen und **Erfahrung** notwendig.
  - Agile Techniken
  - Wie viel Agilität geht "trotzdem" noch?
- **Kreativität** notwendig.
  - Reibungsverluste erfordern konstruktive Ideen

Wann nicht agil?

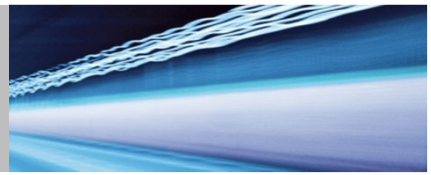


- **Eine agile Vorgehensweise ist nicht immer nützlich!**

➔ Wann nicht?

  - Z.B. wenn **nicht-agile Teams gut „funktionieren“**.
  - Wenn Aufgaben **wenig anspruchsvoll** oder **nicht dringend** sind.
  - Bei stark **eingeschränkten Kommunikationsmöglichkeiten**.

agil = inkrementell denken!

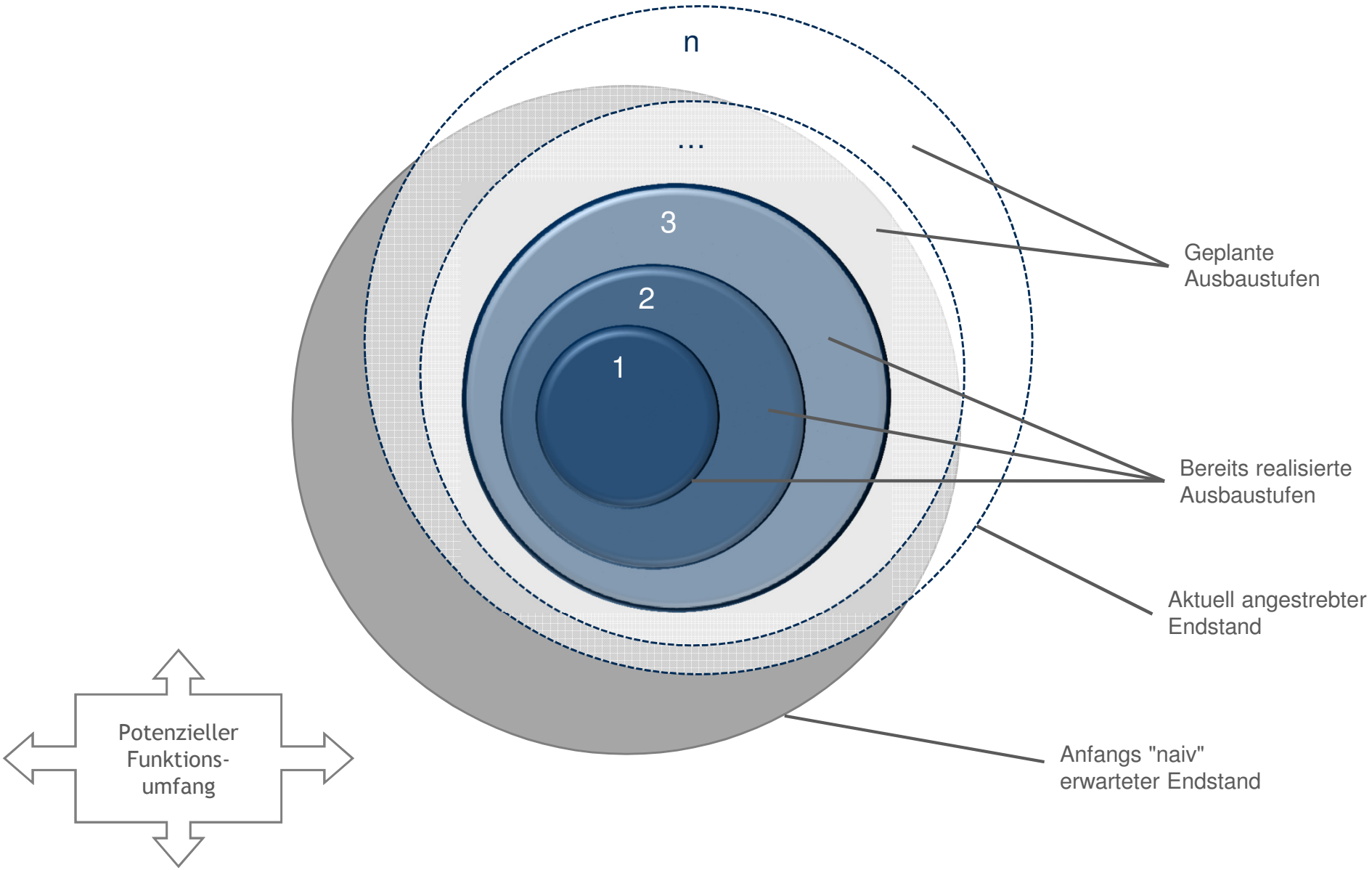


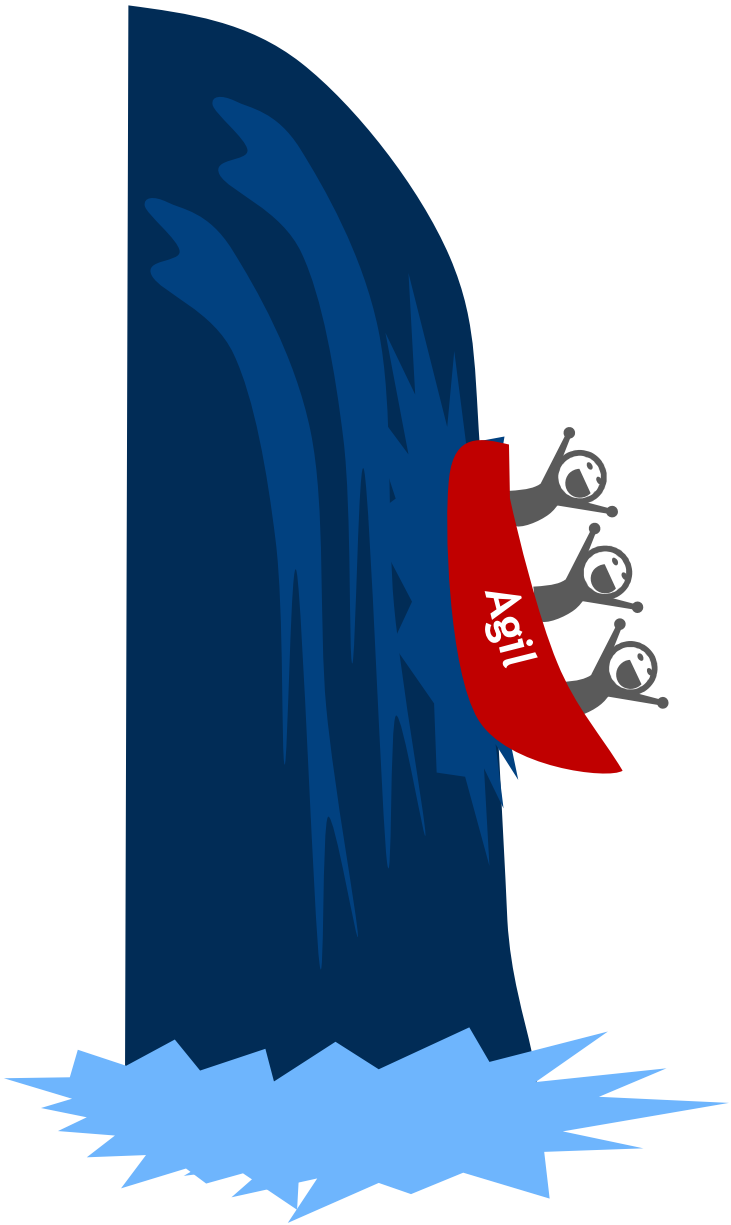
- **Ist Agilität "Projektsache"?**

- Eine **„Inkrementelle Denkweise“** ist wichtig – auch in Fachabteilungen und Management.

➔ Klein beginnen und sukzessive verbessern / skalieren.

Inkremente: Umsetzen von Features in Ausbaustufen

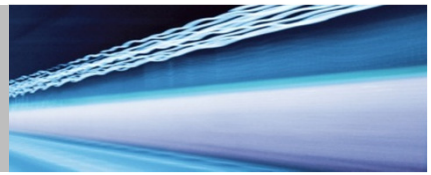




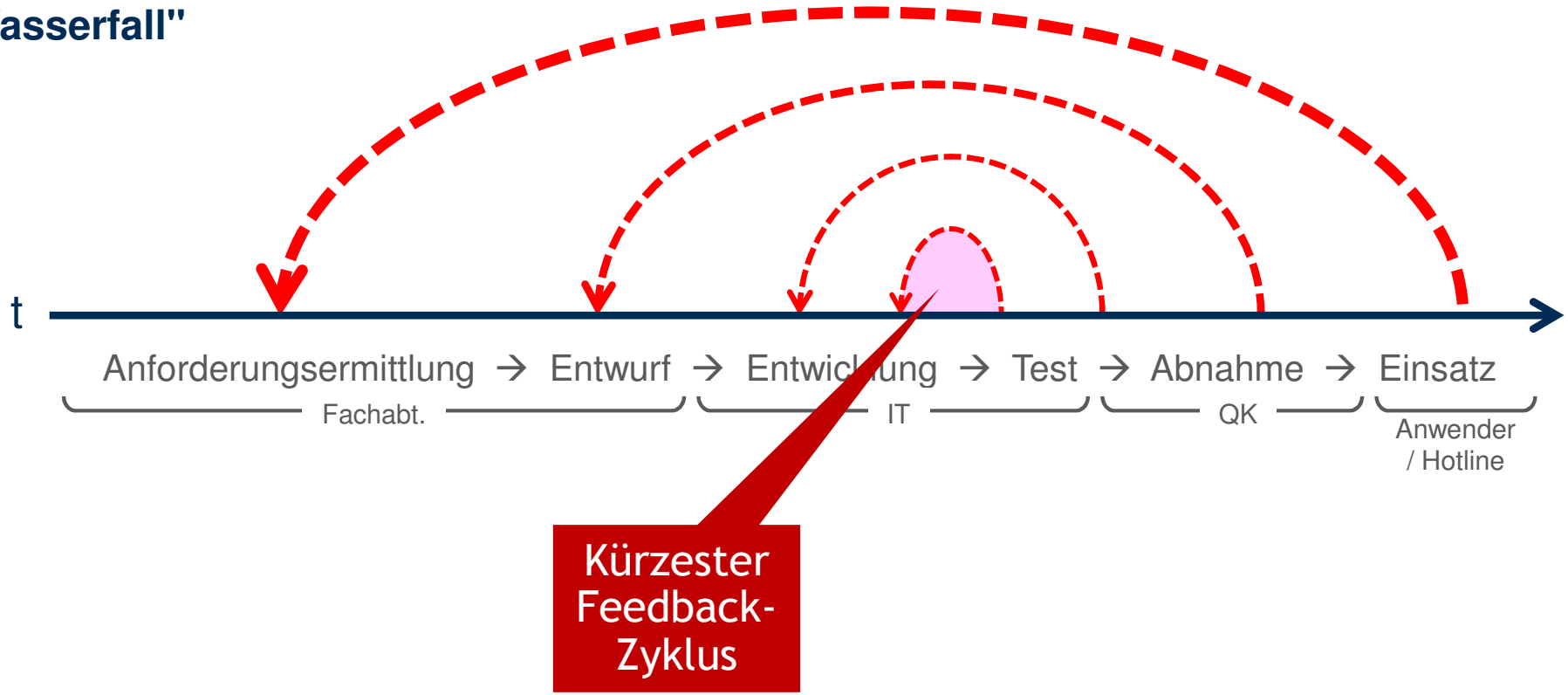
Agiles Vorgehen betrifft das gesamte Unternehmen!

- Im klassischen Wasserfall-Prozess ist selten die Entwicklung problematisch.
- Kritisch: lange Rückkopplungszyklen!
- Ein agiles Entwicklungsteam macht einen „umhüllenden Wasserfall-Prozess“ nicht zu einem agilen Prozess.

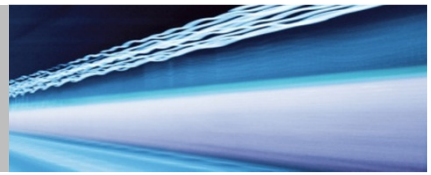
Entwicklung im "Wasserfall"



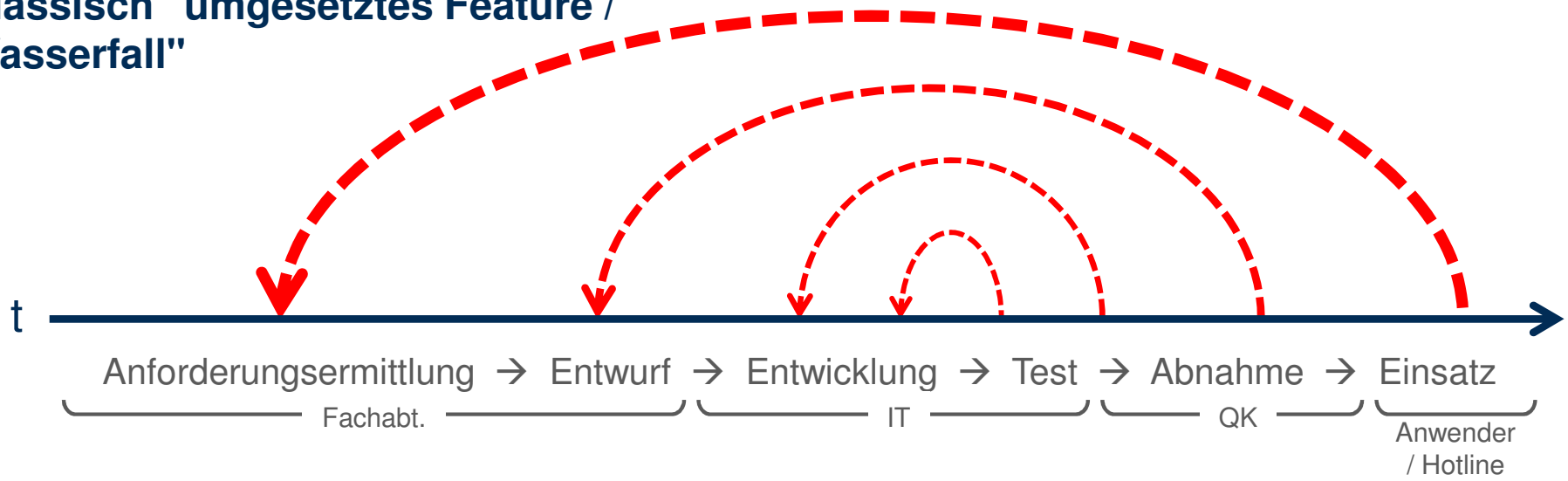
"Klassisch" umgesetztes Feature / "Wasserfall"



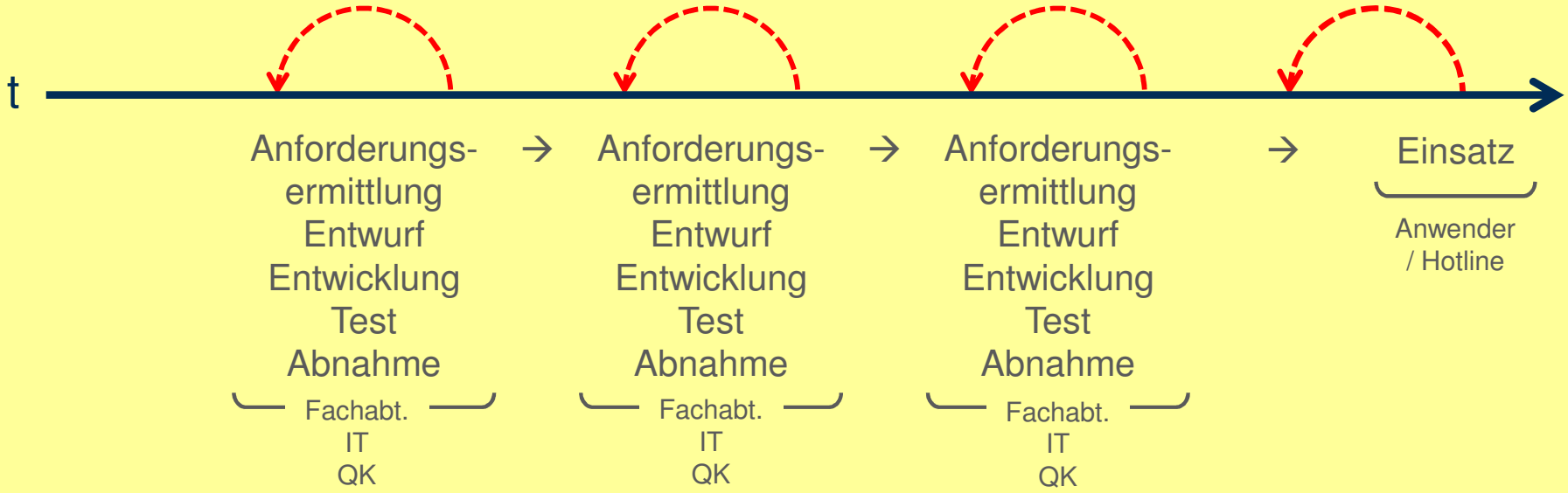
"Wasserfall" vs. iteratives Vorgehen - Feedbackzyklen



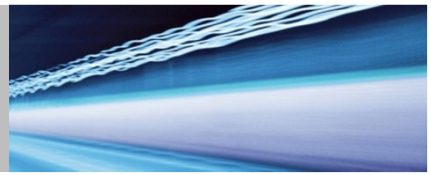
"Klassisch" umgesetztes Feature / "Wasserfall"



Iterativ umgesetztes Feature

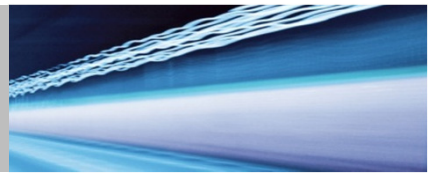


Typische kritische Punkte im Unternehmen



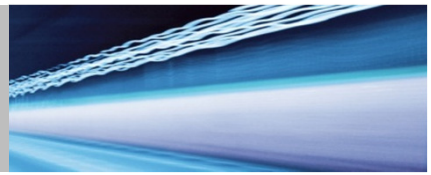
- **Lange Planungshorizonte.** schwergewichtig
- **Historisch gewachsene Unternehmensstrukturen.** starr
- **"Arbeitsteilung".** unkommunikativ
- **Fehlendes Methodenwissen.** unsystematisch
- **Zu wenig Feedback.** intransparent

Planungshorizonte



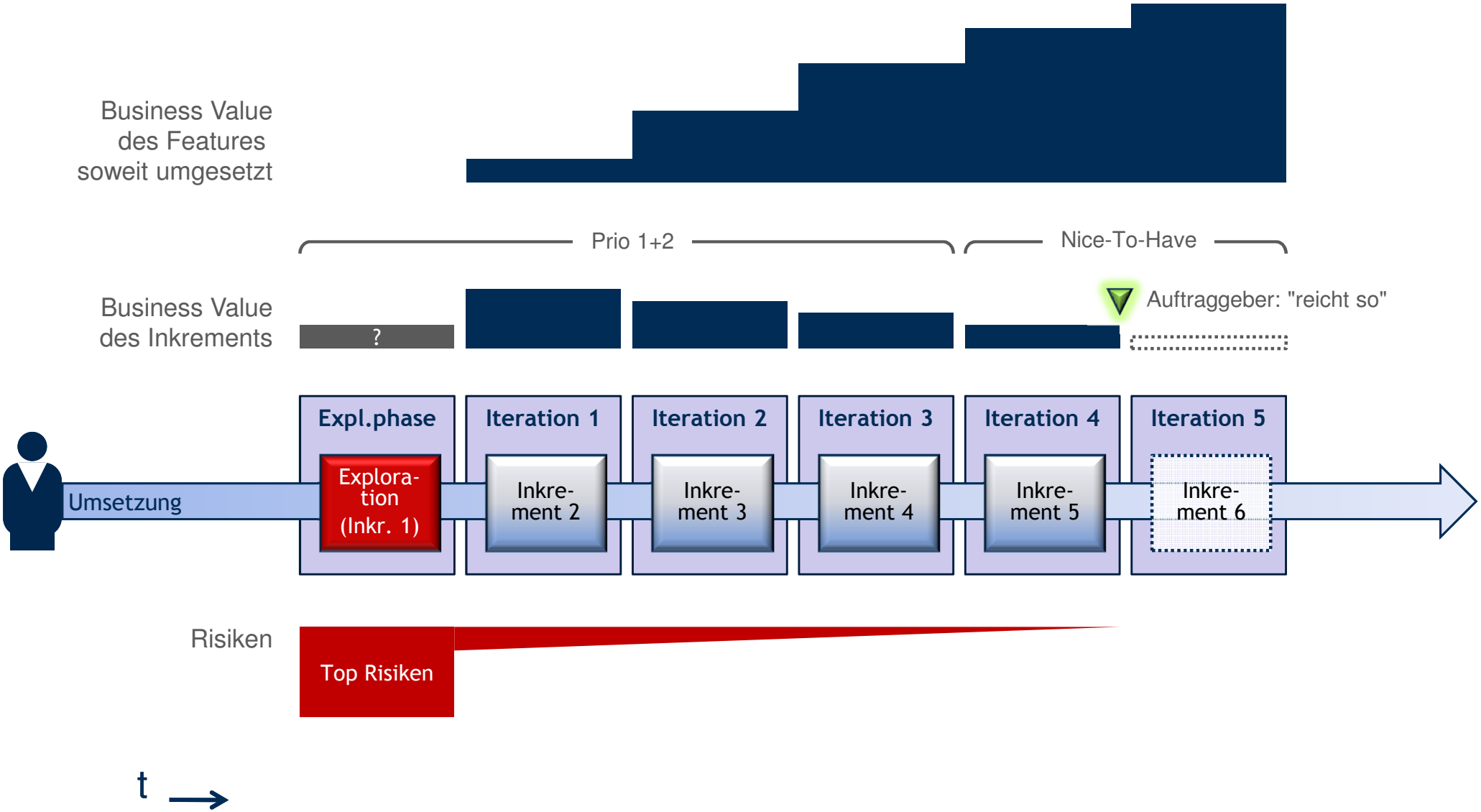
- Der **Planungshorizont in Unternehmen ist oft zu lang**
  - Verbindliche Aussagen zum Ressourcenbedarf sind zu Beginn der Planung nicht möglich.
  - Frühe Abschätzungen durch agile Projekte verbessern.
  - Klassische Pufferschätzungen vermeiden.
  - Stoßrichtung:
    - Wo hat die Planung eine feste Basis?
    - Wie kann die Basis gefestigt werden?

Stories schreiben und priorisieren

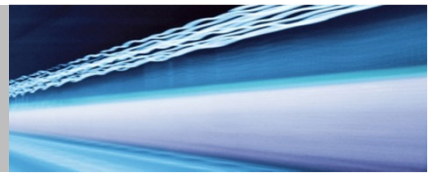


- Differenzierung der Anforderungen:
  kleine, fachlich motivierte Anforderungspakete (Stories)
  - **Priorisieren** und auf das Wichtige fokussieren!
  - Anforderungspakete schätzen (**Aufwand** / **Risiko**).
  - **Unsicherheiten nicht kaschieren**, sondern aktiv angehen, durch...
    - Exploration der Anforderungen oder Technologien.
    - Treffen organisatorischer Entscheidungen.
    - Umpriorisieren, etc.

Business Value / Risikominimierung

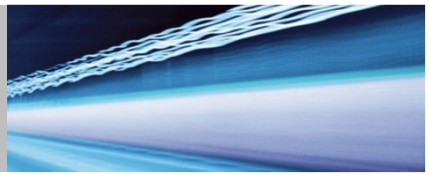


Gewachsene Unternehmensstrukturen



- Unternehmen haben oft **gewachsene Strukturen und Verhaltensmuster**. Dies kann agile Ansätze erschweren:
  - **Alte Systeme** benötigen Refactorings für neue Anforderungen oder Wartung.
  - **Systeme mit hohem Reifegrad** nicht gefährden.
  - **Änderungsbedarf nicht isolierbar**.
  - **Kopfmonopole** verhindern Fokussierung.
  - **Etablierte Arbeitsweisen** behindern agile Techniken.
  - **Geringe Bereitschaft zur Kommunikation**.
  - ...

Beispiel "Kopfmonopole" - Agiler Wissenstransfer



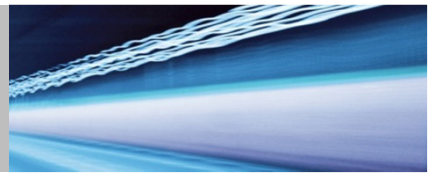
- **Erfahrene Mitarbeiter** besitzen wichtiges Wissen

➔ **Wissenstransfer im Prozess einplanen!**

- Lernende Mitarbeiter mit erfahrenen zusammenbringen.
- Lernende Mitarbeiter können **kleine, inhaltlich in sich abgeschlossene Stories** bearbeiten.
- Erfahrene Mitarbeiter **nehmen diese Stories sorgfältig** ab und geben **Feedback im Detail**.

- Sprachbarrieren und kulturelle Unterschiede im Umgang mit Kritik können den Wissenstransfer stark behindern!

"Arbeitsteilung" in Unternehmen

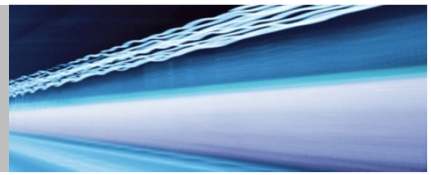


- Beispiele für **arbeitsteilig arbeitende Teams**:
  - Separate Teams für **einzelne Teile der Systemlandschaft**.
  - Separate Teams für **einzelne Aufgabenbereiche**:

    Anforderungsanalyse, UX, Entwicklung, Architektur, QK, Helpdesk, ...

+ Spezialisten verstehen ihr Fach!
+ Sparten- oder System-orientierte Teams arbeiten (für sich betrachtet) effizient!

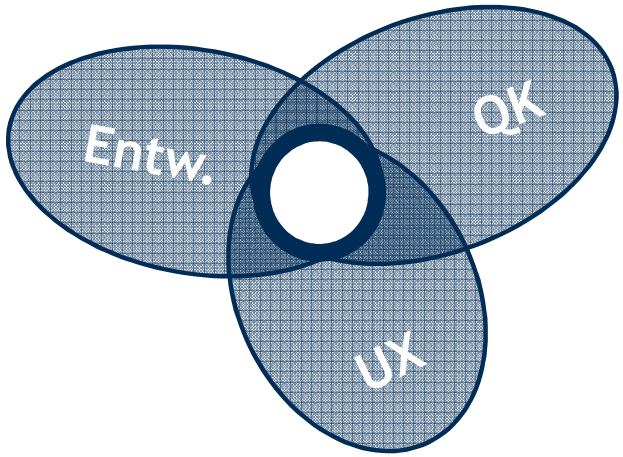
Negative Folgen der Arbeitsteilung



- **"Koordinations-Overhead“** zwischen Projekt und Bereichen (z.B. QK):
  - Missverständnisse, schleppende Abstimmung oder Ergebnistransfer.
  - Arbeitsteilige Stories kaum abschätzbar: Projektrisiko!
  - Verbindliche Planung ist erschwert.
- Regelmäßige **"Wartezeiten"** (z.B. Entwickl. und QK)

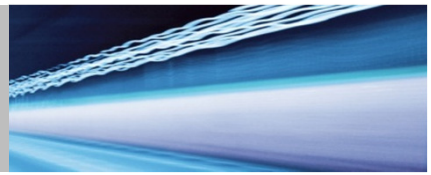
➔ Sind arbeitsteilige Teams (gerade im agilen, stark kommunikativen Kontext) ineffizient?

Ansätze bei Problemen der Arbeitsteilung



- **Integrierte Teams** bilden.
- Mitarbeiter einem Team zu **100%** zuordnen.
- **Räumliche Nähe** (wenn möglich).
- **Tätigkeiten parallelisieren** (z.B. QK und Entwicklung)
- Eine **leichtgewichtige Steuerungsebene** einführen.

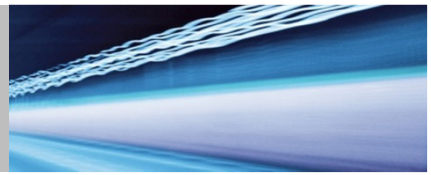
Folklore über Agiles (1)



„Unit-Testing verlangsamt die Entwicklung“

- Unit-getesteter Code enthält **weniger Fehler**.
- Unit-Test-Aufwände sind **planbar**, Bugfixing-Aufwände nicht!
- Unit-Tests führen zu **besseren Schnittstellen**.
- Unit-Testen **verbessert das Verständnis** der Aufgabe
- Unit-Test-Code ist **Beispiel-Code**.
- ...

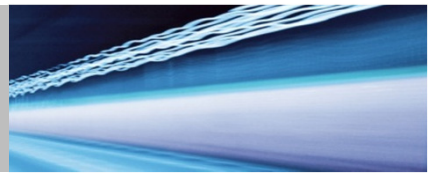
Folklore über Agiles (2)



„Pair-Programming bedeutet Verdoppelung des Aufwands“

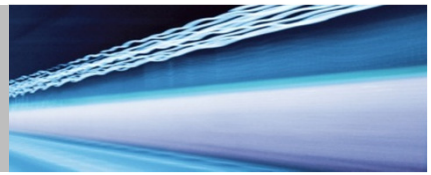
- Ein Pair **versteht seine Aufgabe**.
- **Fehlentwicklungen sind unwahrscheinlich**.
- Ein Pair entwickelt **bessere Lösungen** mit deutlich weniger Fehlern,
- **Ein Pair duldet keinen Leerlauf.**
- Ein Pair findet auch für monotone Arbeiten Lösungen; sonst auch mal aufteilen.
- ...

Folklore vs agiles Methodenwissen



- **Know-How über agile Techniken**, über Funktionsweise und Vorteile von agilen Ansätzen fehlt oft.
  - **Schulungen genügen nicht!**
  - **Positive Erfahrungen sind notwendig!**

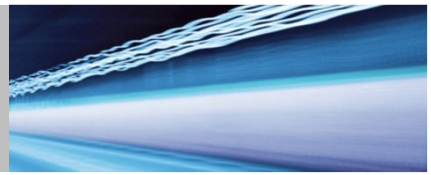
Zu wenig Feedback



- **Feedback über Aufwände** ist schwer zu erhalten:
  - Betriebsrat stimmt Tracking oft nicht zu.
  - Aufwände externer Mitarbeiter erst nach Rechnungsstellung.
  - Auswertung des Controlling selten brauchbar.

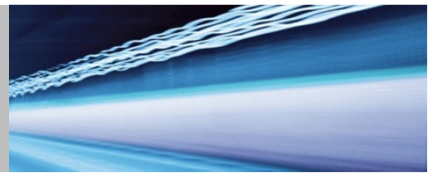
- Aufwände sind oft **nur näherungsweise bestimmbar**
  ➔ Problem für Budgetplanung und Hochrechnungen.

Produkt- und Projektstatus

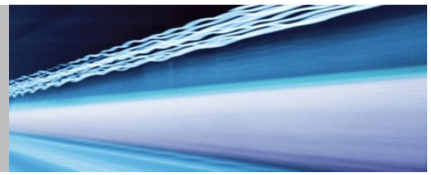


- Der aktuelle Status muss bekannt sein.
  - Statusermittlung von Produkt und Prozesses ist **zeitaufwändig**.
  - Feedback zum Produktstatus erfordert **Produktqualität**.
    ➔ Verlässliches Feedback geht nur auf Basis **erfolgreich abgenommener Stories**.

Reporting



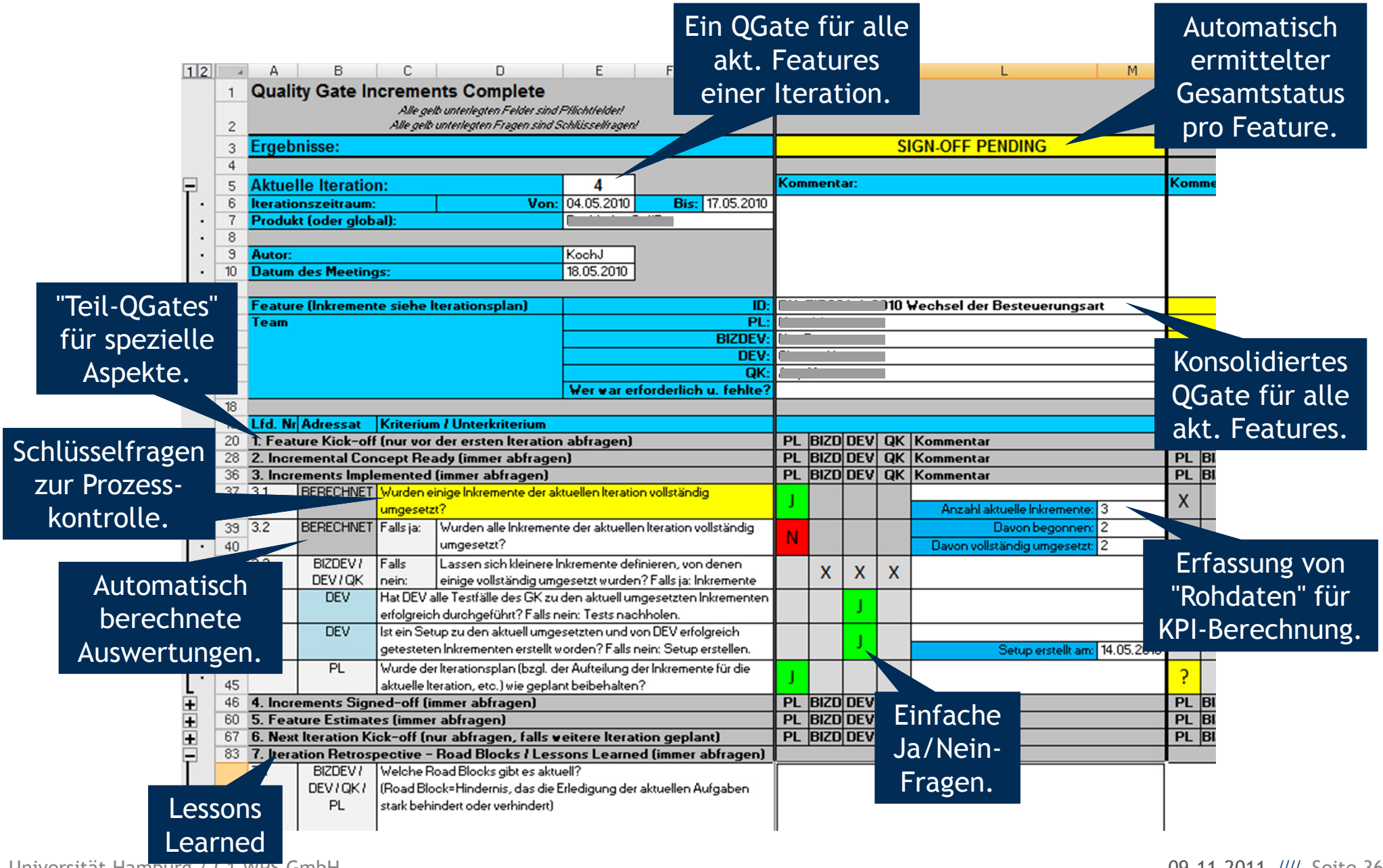
- Reporting liefert **wichtiges Feedback an das Management** zur Zielkontrolle und Koordination.
  - Agile Ansätze betrachten **einzelne Teams** und **interne Feedback-Mechanismen**.
  - Internes Feedback ist **zu detailliert für das Reporting**.



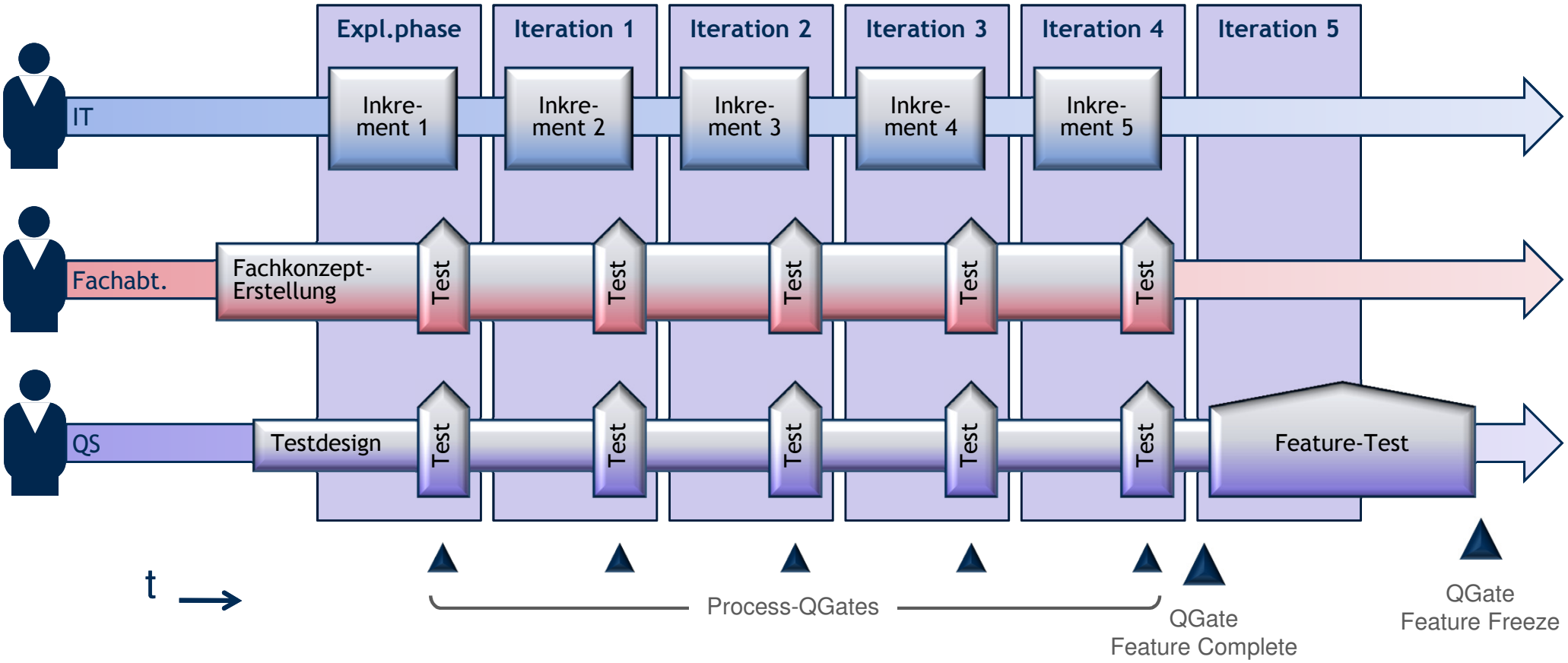
➔ Einfache Kennzahlen:

- Wie viele Stories wurden begonnen?
- Wie viele davon abgeschlossen?
- Wie viele davon getestet?
- Wie viele davon erfolgreich abgenommen?

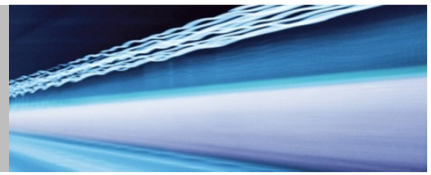
Process-Quality-Gates - Beispiel



Beispiel: Verzahnung von Teilabnahmen & Quality Gates



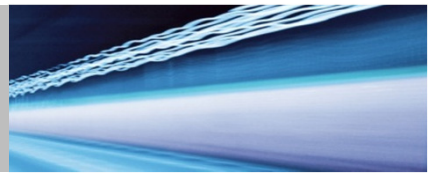
Process-Quality-Gates



+ Kernaspekte des Prozesses werden **verlässlich geprüft**.
+ Kennzahlen helfen gerade unerfahrenen Teams.
+ Sehr gute Grundlage für ein **vergleichbares, Projekt-übergreifendes Reporting**!
+ **Kontrollierbarer Qualitätsstandard**.

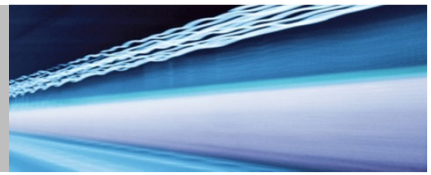
- Ein Process-Quality-Gate ist nur ein Hilfsmittel – Retrospektiven sind weiter wichtig.
- Prozess-Quality-Gates möglichst schlank halten aber regelmäßig durchführen.

Lessons learnt



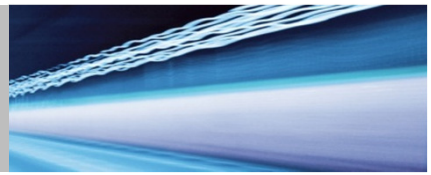
- Zunächst die **Idee des inkrementellen Vorgehens** vermitteln.
  - Taktung in Iterationen kann "leeres Planungsritual" werden.
  - Wichtig: Je Iteration ein **abnehmbares Produktinkrement**.
- Die **Qualität** muss stimmen:
  - Iterationsplanung wird durch Bugfixes hinfällig.
  - **Qualitätsanspruch und funktionierende Qualitätskontrolle** als Grundlage.
  - Ziel: Ein hohes Qualitätsverständnis im Team verankern – keine Bugs tolerieren!

Lessons learnt



- **Begleitendes Coaching** ist unerlässlich.
  - **Vermeidet zu flache Lernkurven** und zu hohes Lehrgeld.
  - Stellt **Team-übergreifendes Verständnis** sicher.
  - **Hilft über die ersten Hürden**
    - Welche Form haben fachliche Aufgabenpakete?
    - Wie gehen wir technische Aufgaben an?
    - Dürfen wir vor Abnahme des letzten Inkrements weiterentwickeln?
    - Wie passt agiles Vorgehen mit Architekturblaupausen zusammen?
  - Hilft, **„Folklore“ über Agiles zu reduzieren.**

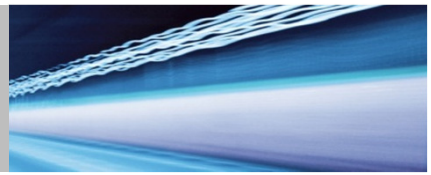
Lessons learnt



- Meilensteinplanung ist in Großprojekten unerlässlich, wenn Teams ein gemeinsames (Zwischen-)Ergebnis liefern sollen.
- **ABER: Auch im Großen agil denken!**
  - Prozessbezogene Meilensteine:
    - Verbindliche Ressourcenabschätzung.
    - Klärung von Unwägbarkeiten.
  - Meilensteine dürfen keine agile Planung verhindern!

- **Auch agile Projekte brauchen klare langfristige Ziele!**
- Ein tragfähiges **Leitbild** für die Produktentwicklung definieren.

Lessons learnt



- **Die Einführung einer agilen Vorgehensweise „dauert“!**

- Agile Techniken werden oft als **Zusatzbelastung** erlebt.
  - Agilität „abgemildert“ einzuführen, kann kontraproduktiv sein.
  - **Tipp:** Statt Iterationen zu verlängern, eher verkürzen.
  - Auf der **korrekten Anwendung** agiler Techniken bestehen:
    - **Saubere, abnehmbare Zwischenstände!**
    - **Kurze Zeitabstände!**