GEBIT Solutions

Die Experten für Java, Anwendungsentwicklung und Requirements Engineering

Anforderungsbasiertes Projektmanagement in der Anwendungsentwicklung

Tilo Sauer

Agenda

- Motivation
- Zwei Thesen
- Drei Projekttypen
- Vier Disziplinen
- Bausteine zur Lösung
- Fazit

Eine Motivation
Das große Ziel...

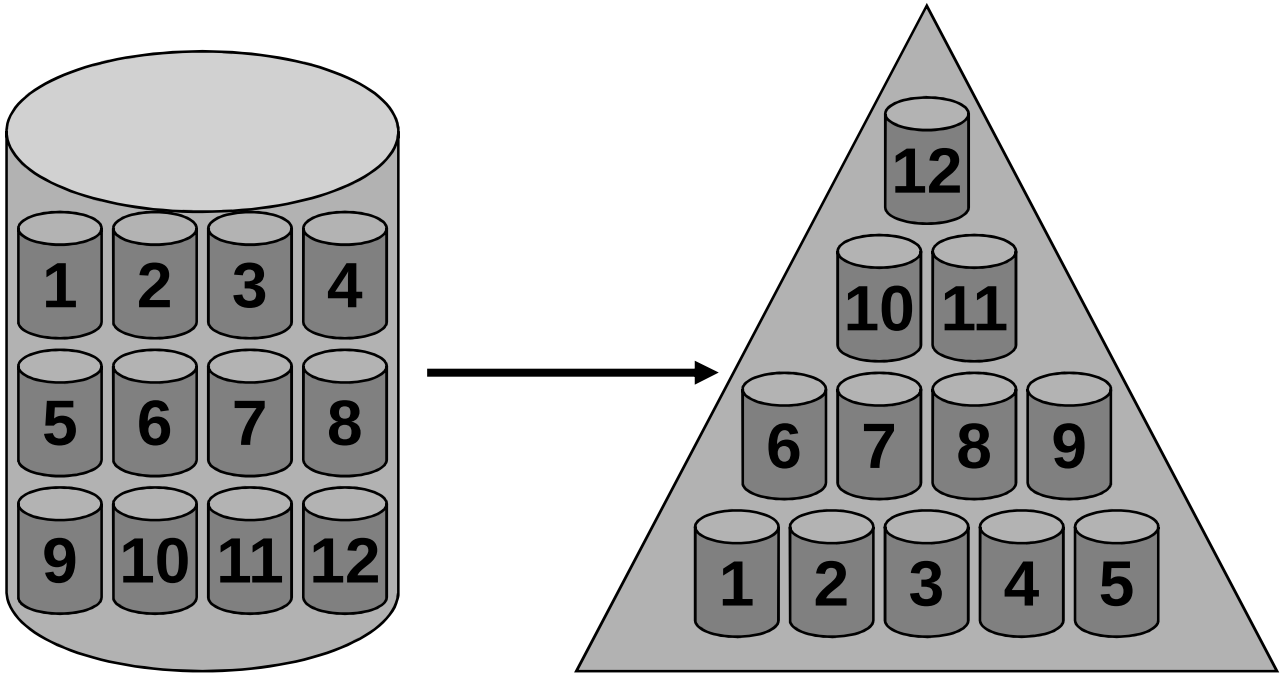
- **Wir wollen die betriebliche Anwendungsentwicklung auf ein Niveau bringen, dass sie die typischen *Versprechen* der Standardsoftware einlösen kann**
  - Klare, dokumentierte Prozesse und Anforderungen
  - Kalkulierbare Budgets
  - Eingehaltene Termine
  - Anpassbarkeit an zukünftige Anforderungen und Technologien
  - Kostengünstig zu betreiben und integriert mit anderen Systemen

Zwei Thesen

1. **Es gibt *notwendigerweise verschiedene* Projekttypen; diese erfordern angepasste Vorgehensweisen**
   - klassisches Wasserfallvorgehen funktioniert bekanntlich selten
   - auch „agil“ passt nicht immer

2. **Es ist eine *ganzheitliche Sicht* über alle Projektdisziplinen hinweg notwendig**
   - Ideen, Anforderungen, Design, Implementierung, Test, Betrieb, Änderungen, ...
   - Das alles muss auf eine stringente – aber handhabbare – Art zusammengebracht werden

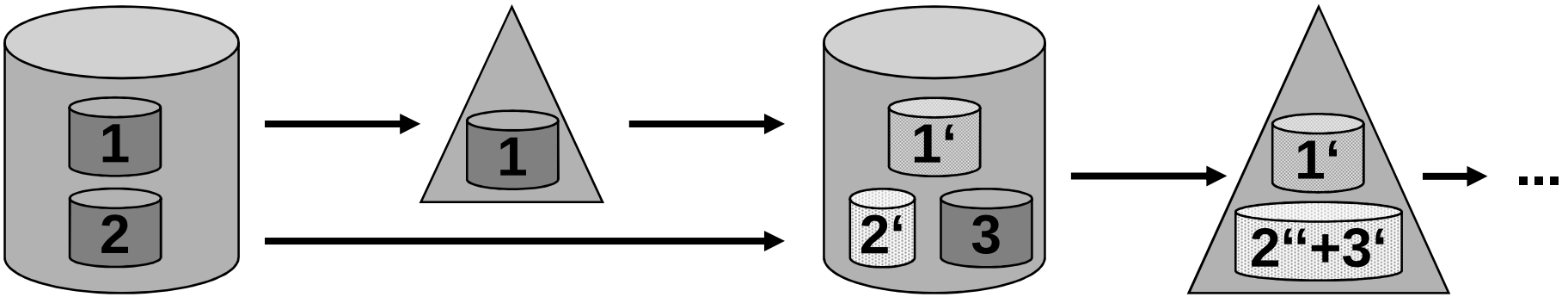
Drei Projekttypen
Typ 1: Statisch, wasserfallartig



Anforderungen **Produkt**

Anforderungen: Umfangreich im Vorfeld erhoben, vollständig und stabil
Budget: Fix – oft als Festpreis / Werksvertrag für den Dienstleister
In der Praxis: CR-Hölle

Drei Projekttypen
Typ 2: Agil, inkrementell

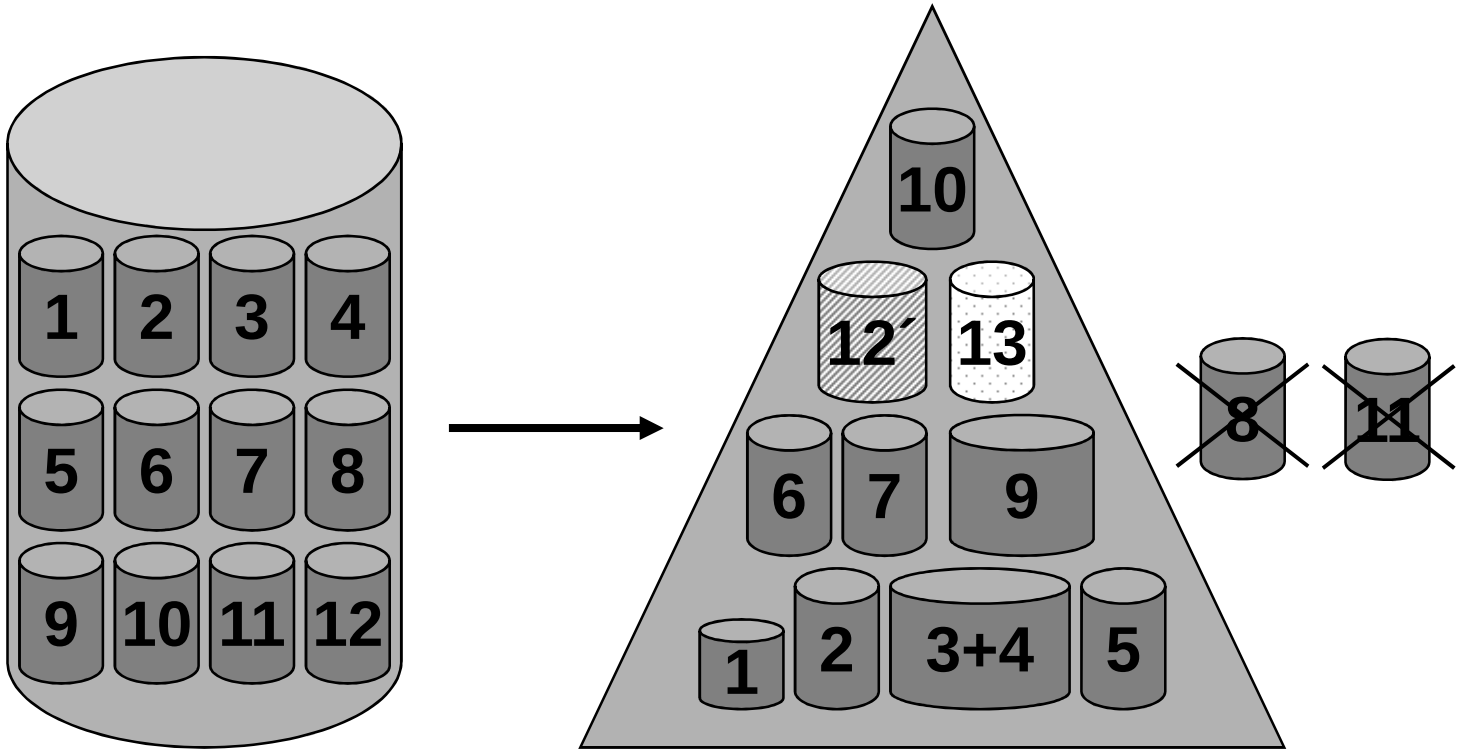


Anforderungen: **Entstehen größtenteils und ändern sich im Laufe des Projekts. Werden explizit nicht sehr detailliert erhoben.**

Budget: **Kann durchaus fix sein. Meist aber als Dienstleistungsvertrag. Scope des Gesamtprojekts ist immer variabel.**

Sie müssen keine klassischen Festpreisprojekte / Werksverträge machen? Herzlichen Glückwunsch! Nehmen Sie diesen Projekttyp!

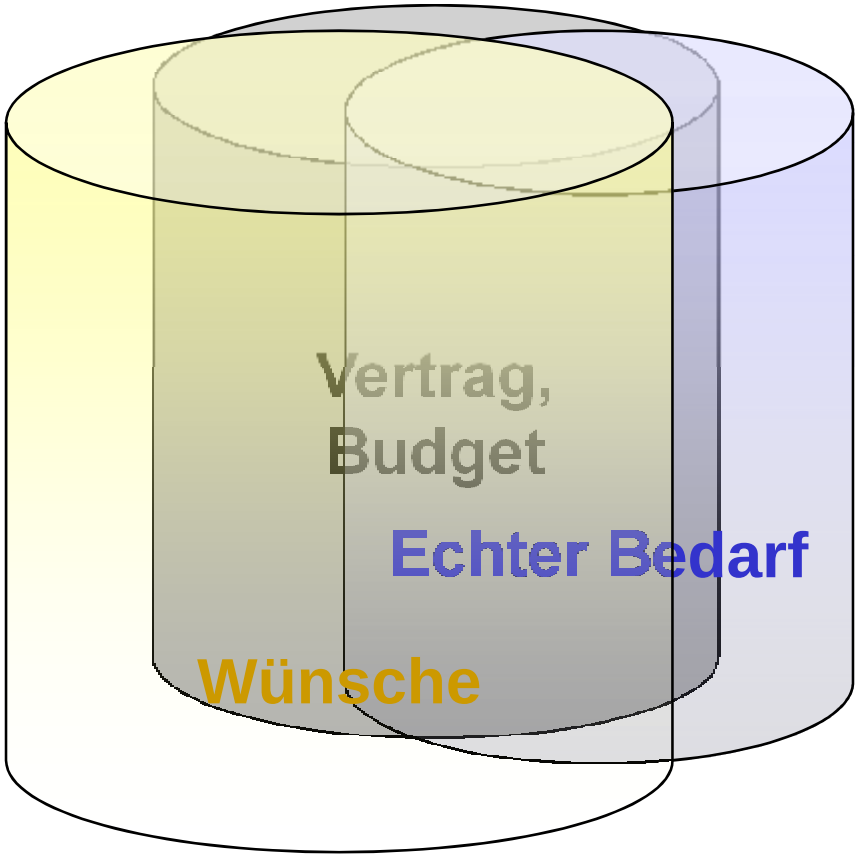
Drei Projekttypen
Typ 3: „Agiles Festpreisprojekt“



Wenn der Kunde einen Festpreis / Werksvertrag ohne CR-Budget will und der Dienstleister nur das Beste für den Kunden will J
... Harakiri?

Agiles Festpreisprojekt
„Was ich *wirklich* brauche, weiß ich erst am Ende“

Agiles Festpreisprojekt
Änderungen zulassen trotz fixem Budget (1)

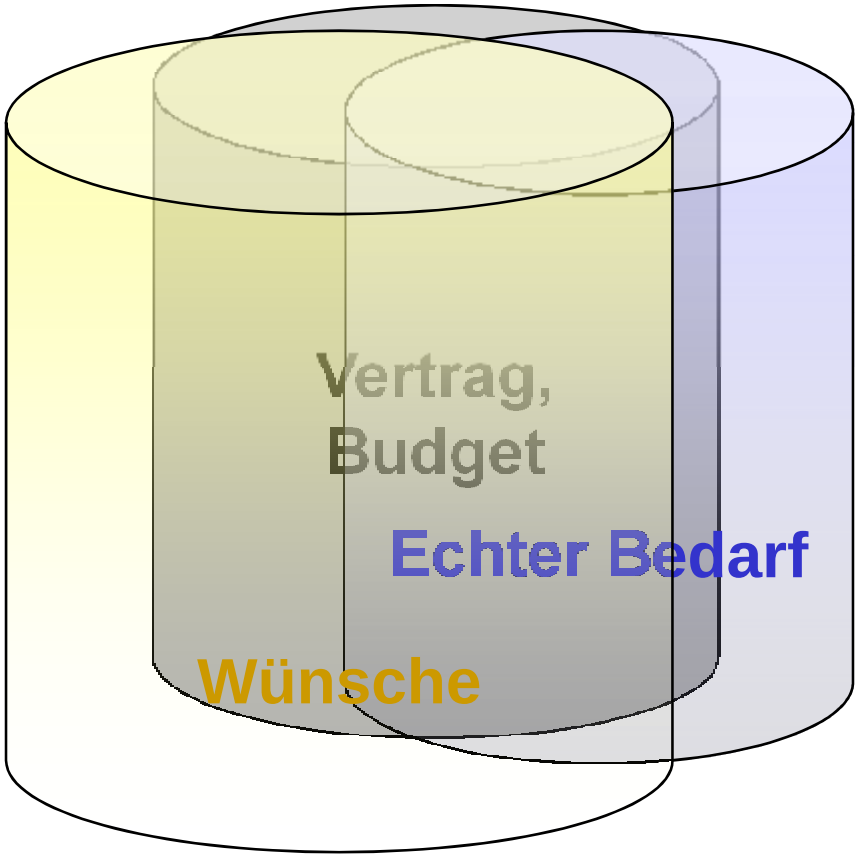


- **Hergestellt werden muss am Ende, was den echten Bedarf deckt**

- **Das Projekt misslingt, wenn...**
  - M strikt der Vertragsumfang umgesetzt wird
  - M jede notwendige Veränderung als kostenpflichtiger CR behandelt wird
  - M jeder Wunsch einfach umgesetzt wird

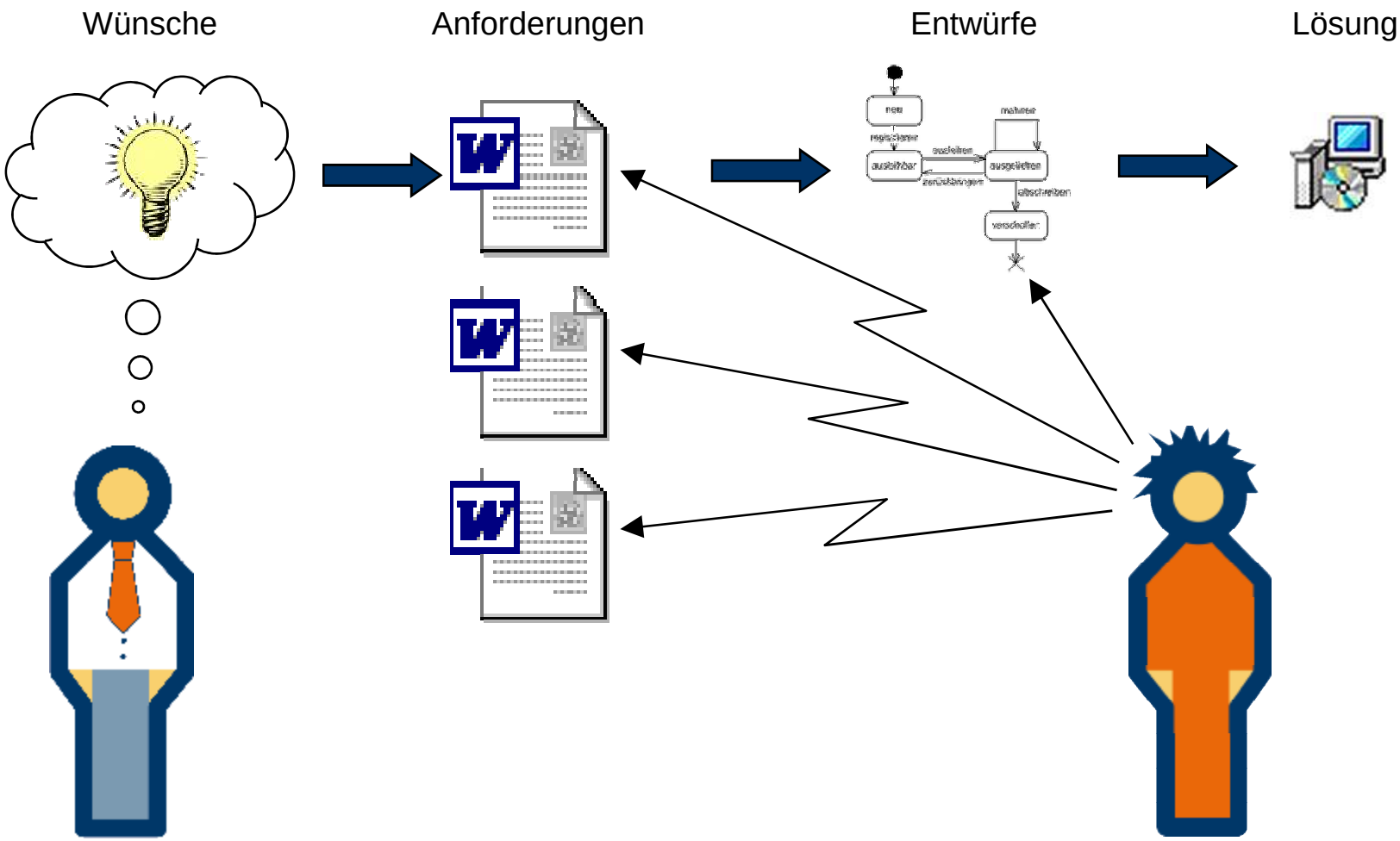
- **Das Projekt gelingt, wenn...**
  - ü der echte Bedarf während des Projekts ermittelt und umgesetzt wird
  - ü dabei trotzdem das Budget eingehalten wird

Agiles Festpreisprojekt
Änderungen zulassen trotz fixem Budget (2)



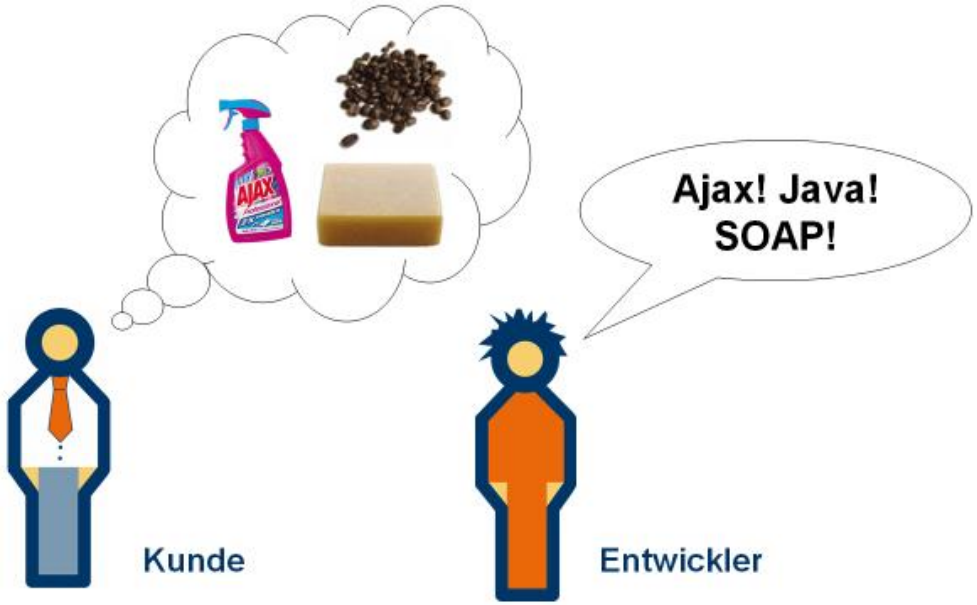
- **Also einfach...**
  - ‚agil' den echten Bedarf im Projekt ermitteln und umsetzen
  - ... dabei ursprünglich geplante Dinge ggf. nicht mehr – oder anders – umsetzen
  - Keine CR's ‚on top' produzieren, sondern geplante Aufwände umverteilen
  - ... dabei Nachvollziehbarkeit und Verbindlichkeit für Alle herstellen
- **Wie ???**

Vier Disziplinen – Bestandsaufnahme
1. Anforderungen und 2. Implementierung



Umgang mit Anforderungen
Typische Probleme

1. **Fehlende gemeinsame Sprache zwischen Entwicklung und Fachbereich**



2. **Kein direkter Bezug zwischen Anforderungen und Implementierung**
   - Dadurch keine Nachverfolgbarkeit (*Traceability*)
   - Projektplanung und Steuerung auf Basis der Anforderungen schwierig
   - Änderungsanforderungen (*Change Requests*) später nur schwer integrierbar

Umgang mit Anforderungen
Weitere Probleme...

3. Zu wenig Struktur in den Anforderungen...

- dadurch kaum Möglichkeiten der formalen Überprüfung
- schwieriges Versionshandling (große monolithische Pakete)

4. ...oder aber: *zu viel* Struktur in den Anforderungen

- bei sehr stark formalisierten Ansätzen: Überforderung der Autoren
- Kleinteilige Ansätze: der Blick für Zusammenhänge geht verloren

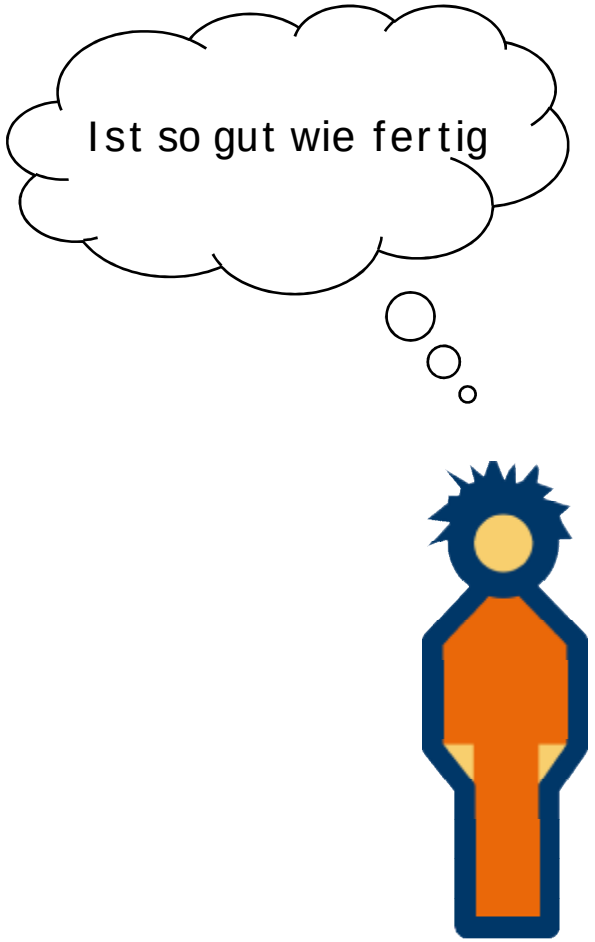
Vier Disziplinen – Bestandsaufnahme
3. Projektplanung

- **Aufwandsschätzung**
  1. Jemand formuliert eine Anforderung
  2. Entwickler schätzt diese direkt **à** 20 PT („das schaffe ich“)
  3. Sein Chef baut Puffer ein **à** 30 PT („Tests, Doku, ... nötig“)
  4. Chef vom Chef baut mehr Puffer ein **à** 40 PT („Integration, Betrieb, ...“)
  5. Vertrieb verkauft das Feature **à** 20 PT („das schaffen wir“)
- **Terminplanung**
  1. Jemand formuliert eine Anforderung
  2. Entwicklungsabteilung analysiert und schätzt nach allen Regeln der Kunst: 500 PT Aufwand, 5 Personen **à** 7 Monate
  3. Management gibt unabhängig davon fixen Termin vor

Vier Disziplinen – Bestandsaufnahme
4. Projekt-Controlling

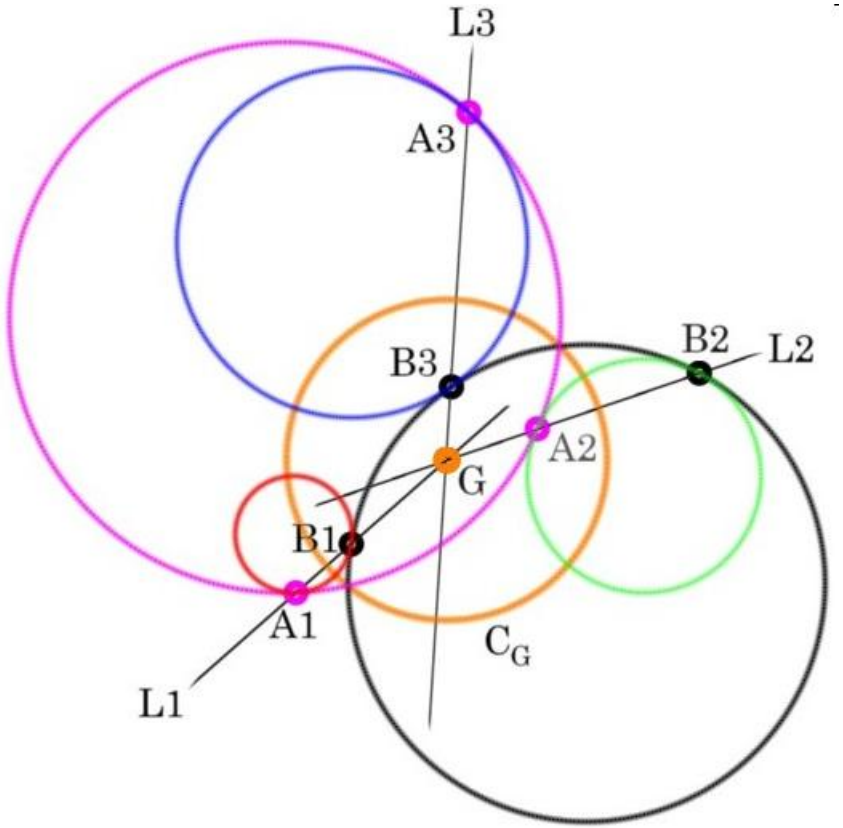
- **Kontrolle des Projektfortschritts auf Grund von Aussagen wie**
  - „Dieser Task ist zu 90% abgeschlossen“ (wochenlang)
  - „Das Framework steht bereits“ (der Endbenutzer sieht nichts)
  - „Wenn dieses Basisproblem gelöst ist, kann die Umsetzung der Fachlichkeit in kürzester Zeit erfolgen“
- **Projektfortschrittskontrolle auf Basis von unscharfen Regelungen**
  - Klare Abnahmekriterien nicht definiert
  - Change Management schlecht in Projektplanung und -Steuerung integriert

Ø **Projektkontrolle und -Steuerung kaum möglich**



Bausteine zur Lösung

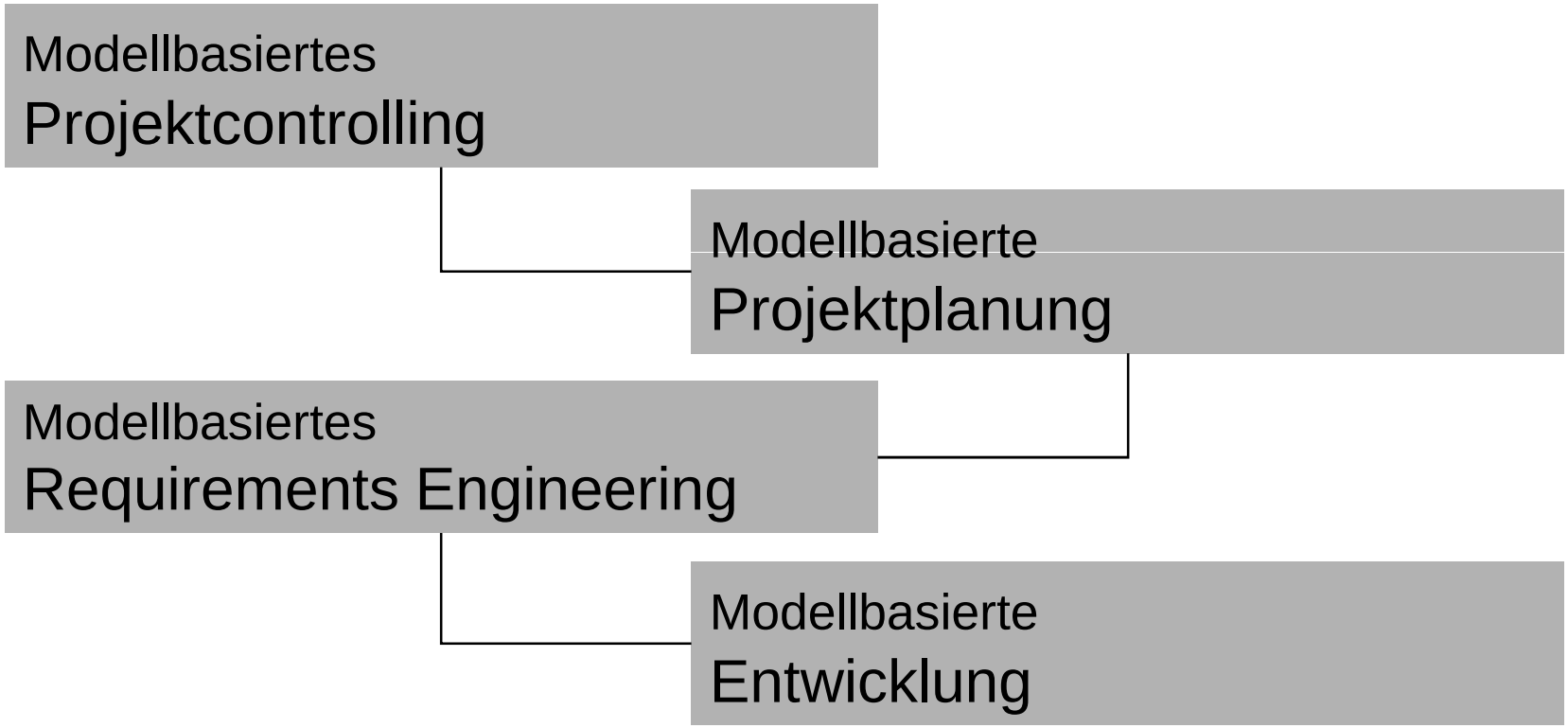
- **Folgende Ansätze haben sich in unseren Projekten in der Praxis bewährt...**



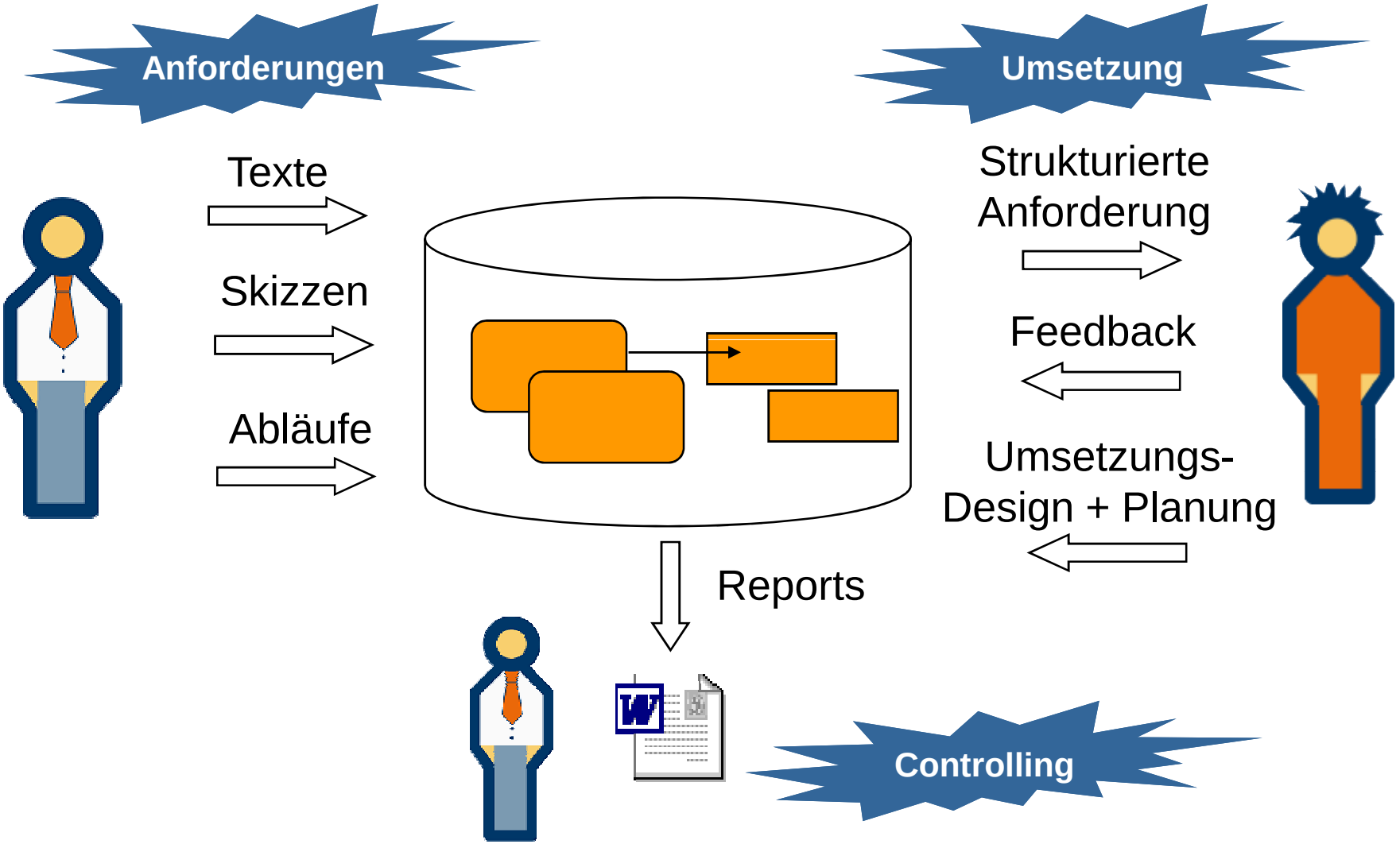
http://upload.wikimedia.org/wikipedia/commons/b/ba/Apollonius_problem_solution_pairs.jpg

Bausteine zur Lösung
Ganzheitliche Sicht herstellen durch Modellbasiertheit

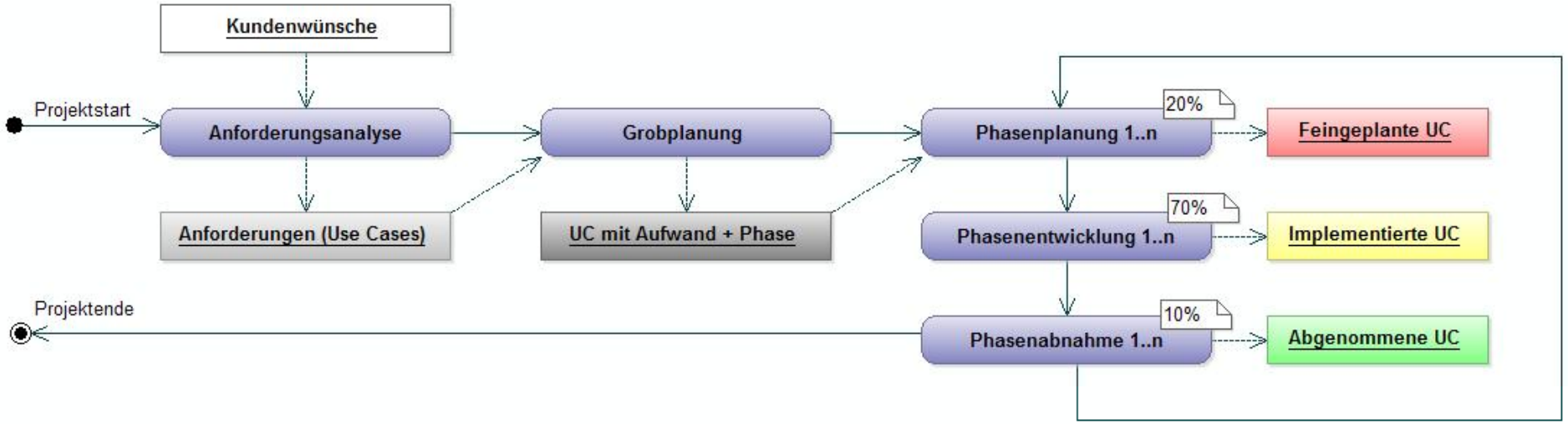
Einsatz von verknüpften Modellen in verschiedenen Disziplinen



Bausteine zur Lösung
Kommunikation über ein gemeinsames Modell



Bausteine zur Lösung
Anforderungsbasiertes Projektvorgehen

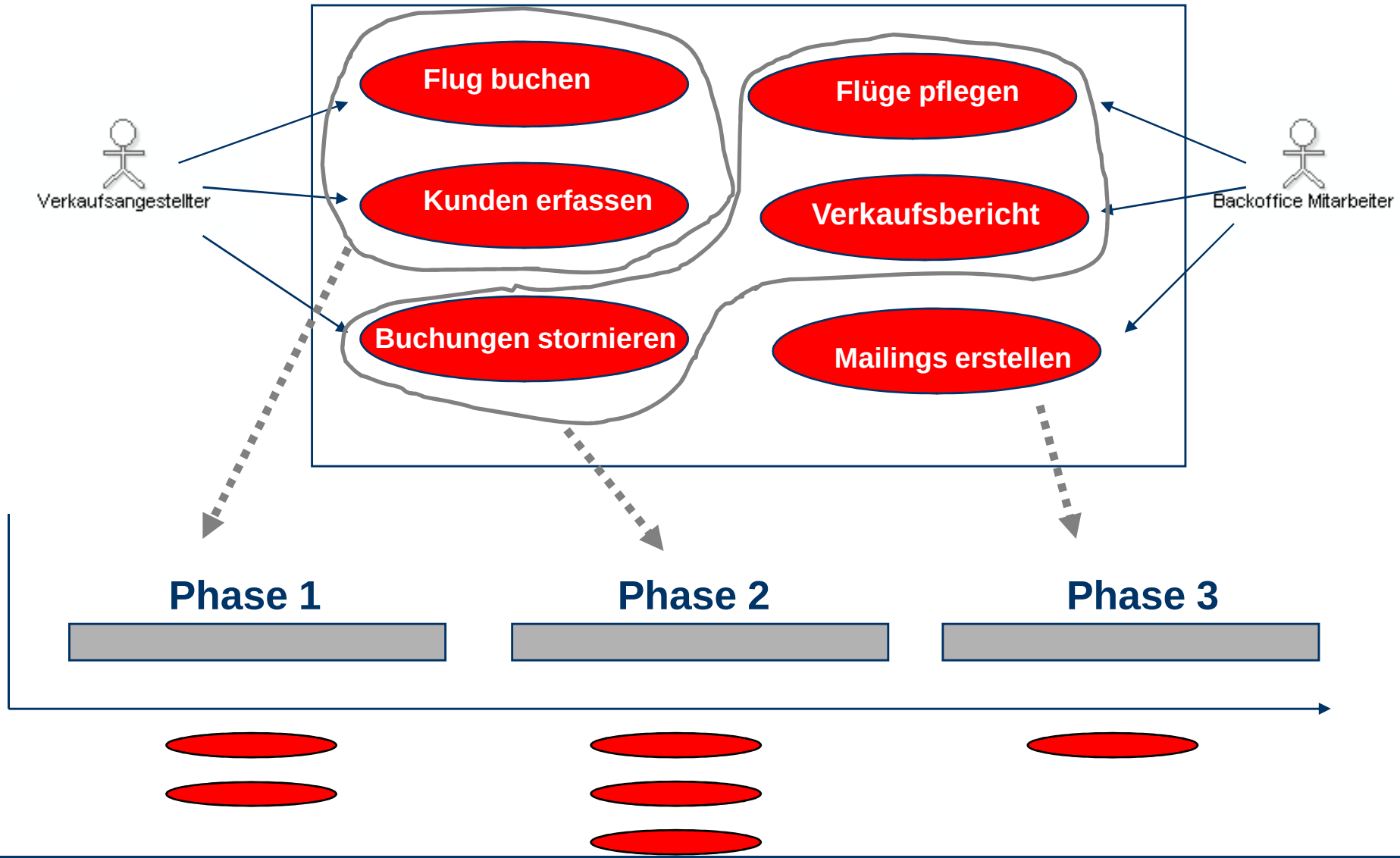


- **Anforderungsanalyse**
  - Kundenwünsche **à** Anforderungen
  - Primär als Use Cases
- **Grobplanung**
  - Use Cases mit geschätztem Aufwand versehen
  - Fachliche Abhängigkeitsanalyse **à** Initiale Zuordnung zu Projektphasen
- **Phasen; jeweils...**
  - Zerlegung der Use Cases, Zuweisung an Teams
  - Entwicklung und Abnahme

Bausteine des Projektvorgehens
Anforderungsanalyse

- **Die *Kundenwünsche* liegen in unvorhersehbarer Form vor**
  - Texte, Tabellen, Bilder, Altsystem, ...
  - Einfrieren dieser Originale – nicht auf dieser Basis weiter arbeiten
- **Vollständige Umwandlung in *Anforderungen* notwendig**
  - primär in Form von Use Cases (funktionale Anforderungen)
    - alternativ: z.B. User Stories in Kombination mit Scrum
  - ergänzt um nicht-funktionale Anforderungen
  - nur noch diese Anforderungen sind die weitere Basis; auch der Kunde muss dann mit diesen weiterarbeiten
- ***Wie* macht man das genau?**
  - eigenes Thema (Requirements Engineering) – wird hier nicht vertieft

Bausteine des Projektvorgehens
Grobplanung



Bausteine des Projektvorgehens
Grobplanung

- **Einzelner Use Case (Grobplanung)**
  - Beschreibt eine für den Endbenutzer wertvolle, überprüfbare fachliche Funktion
  - Mit Vor- und Nachbedingungen, normalem Ablauf, alternativen Abläufen, einzuhaltenden Geschäftsregeln, etc.
  - Also die „reine Lehre“ – und hier nicht weiter Thema
- **Aufwandsschätzung pro Use Case**
  - Richtwert: je nach Erfahrung maximal 15-20 PT direkt schätzen
    - darüber fachlich sinnvoll aufteilen und Einzelteile separat schätzen
  - Enthält Aufwand für
    - Fachliche Abstimmung während der Phasenplanung und Dokumentation des Entwurfs
    - Umsetzung (Modellierung, Coding) inkl. Dokumentation und Entwicklertests
  - Möglichkeiten der Absicherung
    - Unabhängige und verdeckte Vergleichsschätzung durch mehrere Personen
    - Ist-Aufwände zu ähnlichen Use Cases aus vergangenen Projekten heranziehen
    - Spezielle Use Cases durch Experten schätzen lassen
    - Späteres Implementierungsteam einbeziehen, falls möglich (Planungspoker, u.a.)

Bausteine des Projektvorgehens
Grobplanung

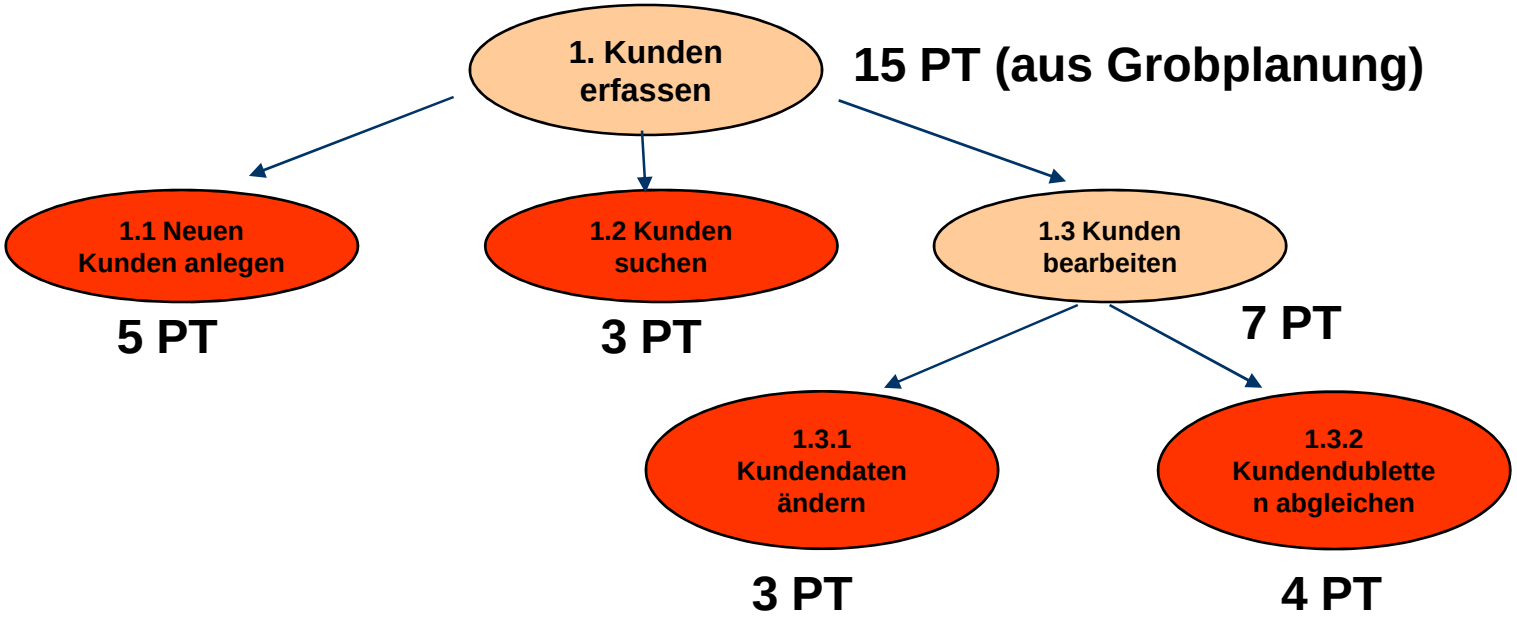
- **Gesamtaufwand ist nun abgeschätzt**
- **Noch variabel: Teamgröße, Phasen, Laufzeit**

- **Unsere „aktuellen Grenzwerte“ für dieses Verfahren...**
  - Teamgröße: 3 bis 20
  - Anzahl Phasen: 3 bis 6
  - Phasenlänge: 1 (kleine agile Projekte) bis 4 Monate (Großprojekte)

- **Wenn man diese Grenzwerte akzeptiert, folgt daraus**
  - Projektlaufzeit: Minimal 3 Monate, maximal 24 Monate
  - Aufwand: Minimal ca. 150 PT, maximal ca. 9.000 PT
  - D.h. darunter und darüber spielt sich das Verfahren *möglicherweise* nicht unbedingt sinnvoll ab

- **Ergebnis: Rahmen für Gesamtprojekt (Aufwand, Zeit, Ressourcen)**

Bausteine des Projektvorgehens
Phasenplanung



- **Fachlichkeit im Detail abklären**
  - Use Cases und begleitende Dokumente ergänzen
- **Große Use Cases in kleinere unterteilen; ggf. umstrukturieren**
  - Ideal: maximal 5 PT Aufwand pro Use Case – leider nicht immer machbar
  - Gesamtaufwand bleibt gleich, wird nur verteilt

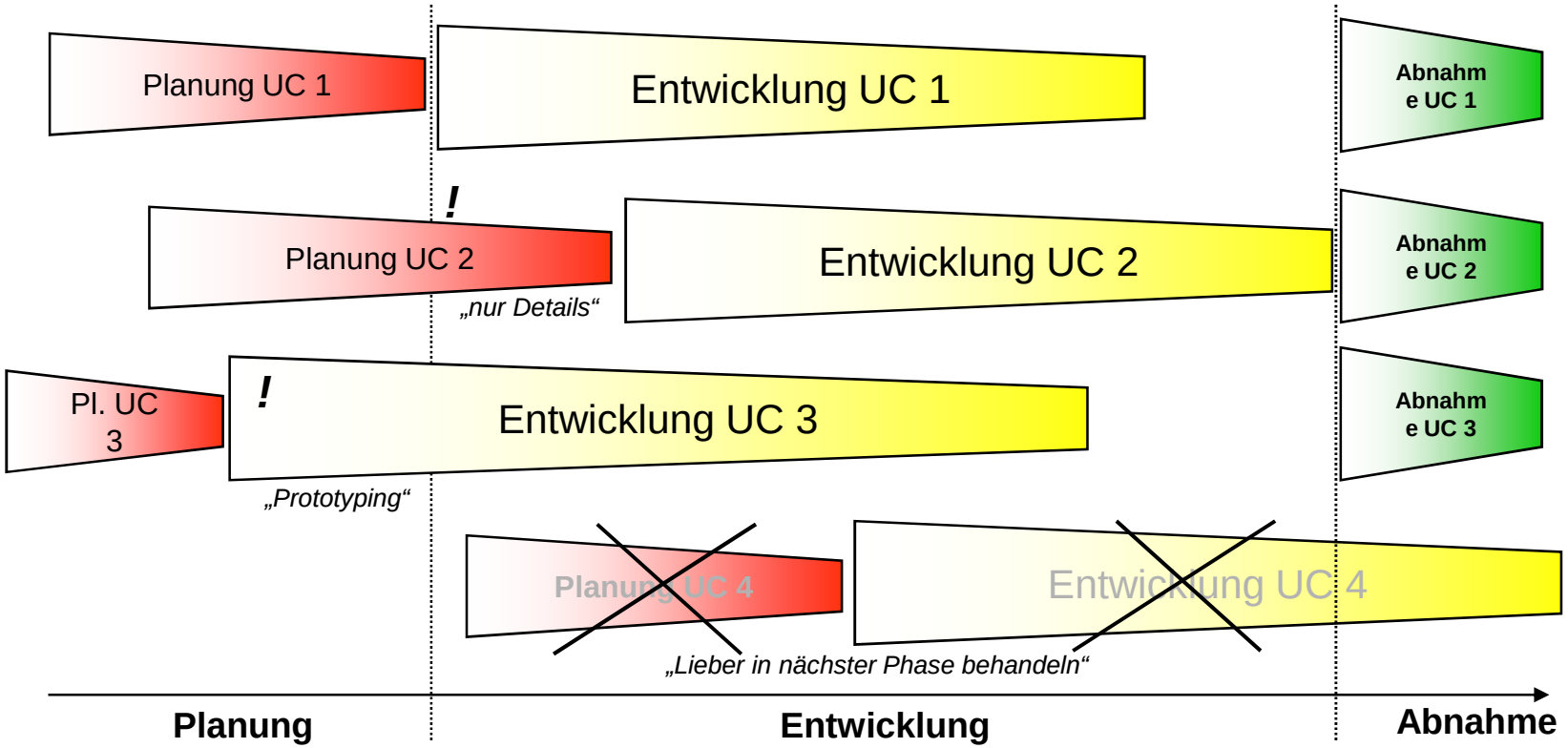
Bausteine des Projektvorgehens
Phasenplanung

- **Einzelner Use Case (Phasenplanung)**
  - **Ziel: *immer noch* eine für den Endbenutzer wertvolle, überprüfbare fachliche Funktion**
  - Auch oder gerade bei „nicht-funktionalen Use Cases“ oder „Framework-Features“: vorher klare Kriterien festlegen, wann dieser Use Case als fertig gelten soll

- **Don‘t**
  - Keine weitere Aufteilung, die nicht fachlich sinnvoll ist, nur um das Ziel „Aufwand <= 5 PT“ zu erreichen

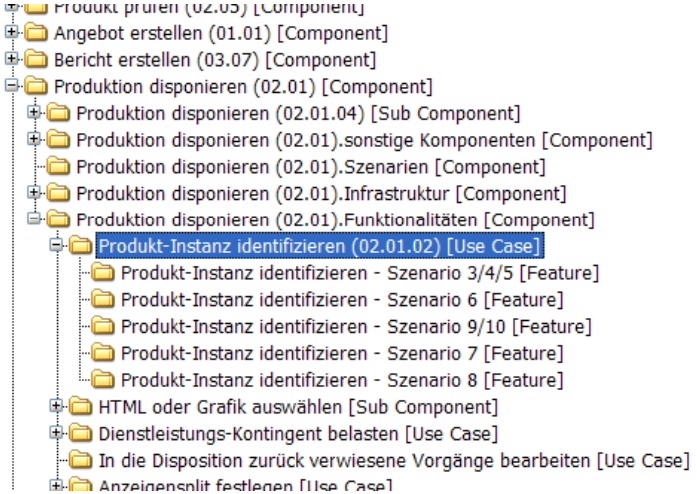
Bausteine des Projektvorgehens
Phasenentwicklung

- **Übergang Phasenplanung à Phasenentwicklung verläuft in der Praxis oft nicht lehrbuchmäßig**
  - ist handhabbar bzw. akzeptabel, sofern die Auswirkungen begrenzt sind



Bausteine des Projektvorgehens
Produkthierarchie

- **Ein Sack voller Use Cases?**
  - Nein: Eine Hierarchie ist sehr nützlich, z.B.
  - Produkt
    - ↳ Komponente
      - ↳ Use Case
        - ↳ Feature



- **Diese Produkthierarchie kann ein zentrales Element für das Projekt sein**
  - stellt die aktuelle Struktur des entwickelten Produkts dar
  - liefert die Anker für Planung, Controlling und Team-Kommunikation
  - kann / wird sich im Laufe der Zeit ändern und erweitern
  - ist das, was „am Ende des Projekts übrig bleibt“

Bausteine des Projektvorgehens
Definition of Done

- „Definition of Done“ – wann ist ein Use Case fertig?
  - Entwickler *beendet die Arbeit* an dem Use Case in dieser Phase
  - Es wird *kein weiterer Aufwand* in diesen Use Case gesteckt
  - Die implementierte Funktion geht dann *genau so* in die Phasenabnahme, um dort endgültig bestätigt zu werden
- Voraussetzung
  - Zugehöriges Use Case Dokument ist vor Implementierung finalisiert / abgenommen
- Fertigstellungskriterien für Use Cases
  - Fachvertreter testet *innerhalb der Phase*, sobald ein Use Case umgesetzt wurde: Er verifiziert dadurch noch einmal die Anforderungen und bestätigt vor allem die Art der Umsetzung
  - Technische Kriterien
    - Von Kollege / technischem Architekt oder teilweise automatisch überprüft
    - Unit-Test (falls sinnvoll) vorhanden und erfolgreich
    - Automatischer GUI-/Regressionstest (falls sinnvoll) vorhanden und erfolgreich
    - Modell-/Code-Dokumentation in guter Qualität vorhanden, Style-Guides eingehalten
    - Verlinkung hergestellt zwischen Use Case (Modell / Dokument) und Design-/Implementierungsartefakten (Diagramm / Source)

Bausteine des Projektvorgehens
Phasenabnahme

- **Jede Phase wird wie ein ganzes Projekt geführt**
  - daher gibt es am Ende auch einen integrativen Test + Abnahme
  - nach der letzten Phase ggf. noch etwas ausführlicher

- **Im Prinzip sollte es keine Überraschungen geben**
  - alle Funktionen / Use Cases wurden bereits in der Phase gesichtet
  - „Natürlich“...
    - ... wird es trotzdem Bugs und Änderungsbedarf geben
    - ... hat man sich dies und das im Zusammenspiel anders vorgestellt

Bausteine des Projektvorgehens
Projektstatus aus Use Case Status

- **Projektstatus = Fertiggestellte + abgenommene Use Cases**
  - halb fertig oder „90%“ zählt nicht
  - Als Ertragswert kann z.B. die Plan-PT Summe der implementierten bzw. schon abgenommenen Use Cases dienen
- **Mögliche Statuswerte:**
  - **geplant** — 

    Use Case ist für eine Phase geplant („To-do“)
  - **implementiert** — 

    Use Case ist fertig (*done*) gemäß der vereinbarten Kriterien
  - **abgenommen** — 

    Use Case ist abgenommen
  - **nicht geplant** — 

    Use Case ist noch nicht für eine Phase geplant
  - zurückgestellt
  - gestrichen

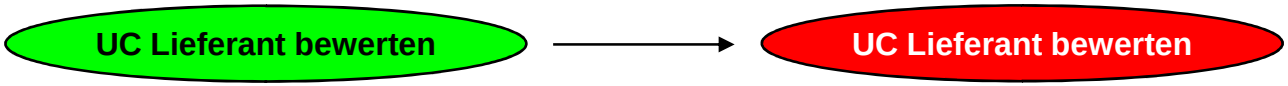
  Use Case ist nicht im Scope

Bausteine des Projektvorgehens
Umgang mit Use Case Inkrementen

- **In einer Phase wird ein Use Case vollständig implementiert und abgenommen**

  - Basis ist die versionierte UC-Beschreibung zum Zeitpunkt der Phasenplanung
  - Statusübergänge innerhalb der Phase nur „nach vorne“
  - Die Anforderungen an den UC können während der Entwicklung erweitert oder geändert werden – aber ohne Einfluss auf die laufende Entwicklung

- **Falls nötig, kann der UC in einer nächsten Phase erneut eingeplant werden**

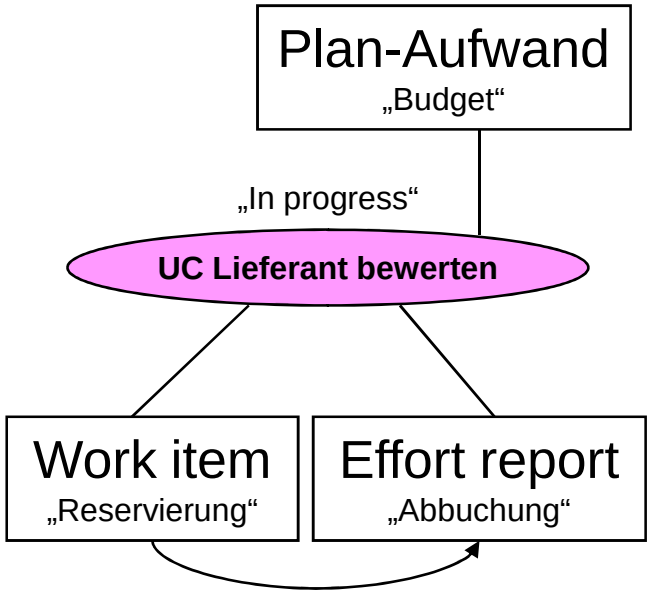


  - Der UC ist in dieser Phase dann wieder „Baustelle“
  - Nachhalten des Werdegangs über Versionskontrollsystem
  - Je nach Bedarf Speicherung historischer Aufwandswerte

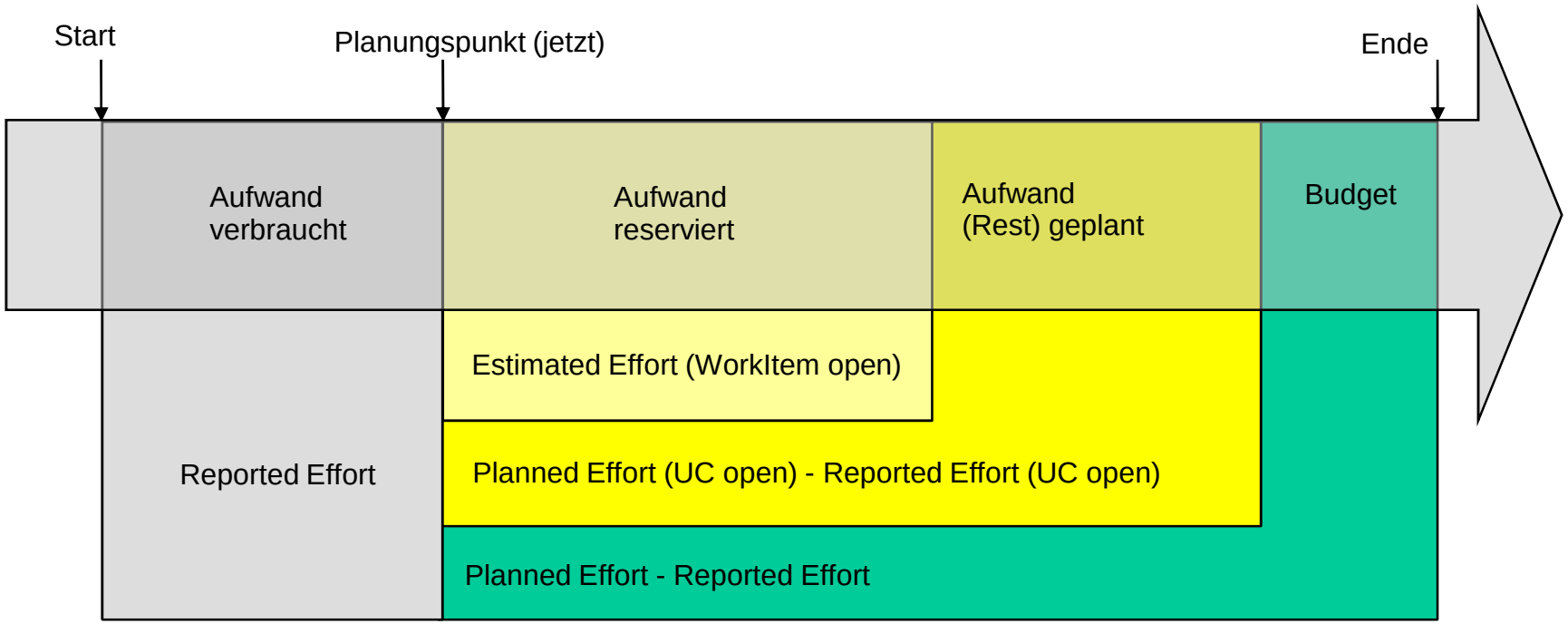
Bausteine des Projektvorgehens
Tracking von Ist-Aufwänden und Prognosen

- **UC-Status liefert klare Aussage: „Das haben wir jetzt“**
- **Nicht direkt abzuleiten sind hingegen:**
  - „Daran arbeiten wir gerade“
  - „So viel haben wir bislang hier investiert“
  - „Diese Dinge sind dafür noch zu tun“
  - „Wir bleiben im geplanten Aufwand“
- **Lösung: Ergänzen des Ansatzes um**
  - „In Arbeit“ Status (*In progress*)
  - Aufwandsmeldungen (*Effort reports*)
  - To-Do Liste (*Work items*)

  am einzelnen Use Case
- **Kann auch Forderung nach <= 5 PT Aufwand pro Use Case relativieren**
  - „Pseudo Use Cases“ werden vermieden



Agile Festpreisprojekte
Aufwandscontrolling (Kostenbetrachtung)



Agile Festpreisprojekte
Umgang mit starren Abnahmekriterien

- **Problem: Abnahmekriterien laut ursprünglichen Anforderungen**
  - Notwendige Änderungen im Projekt werden erschwert

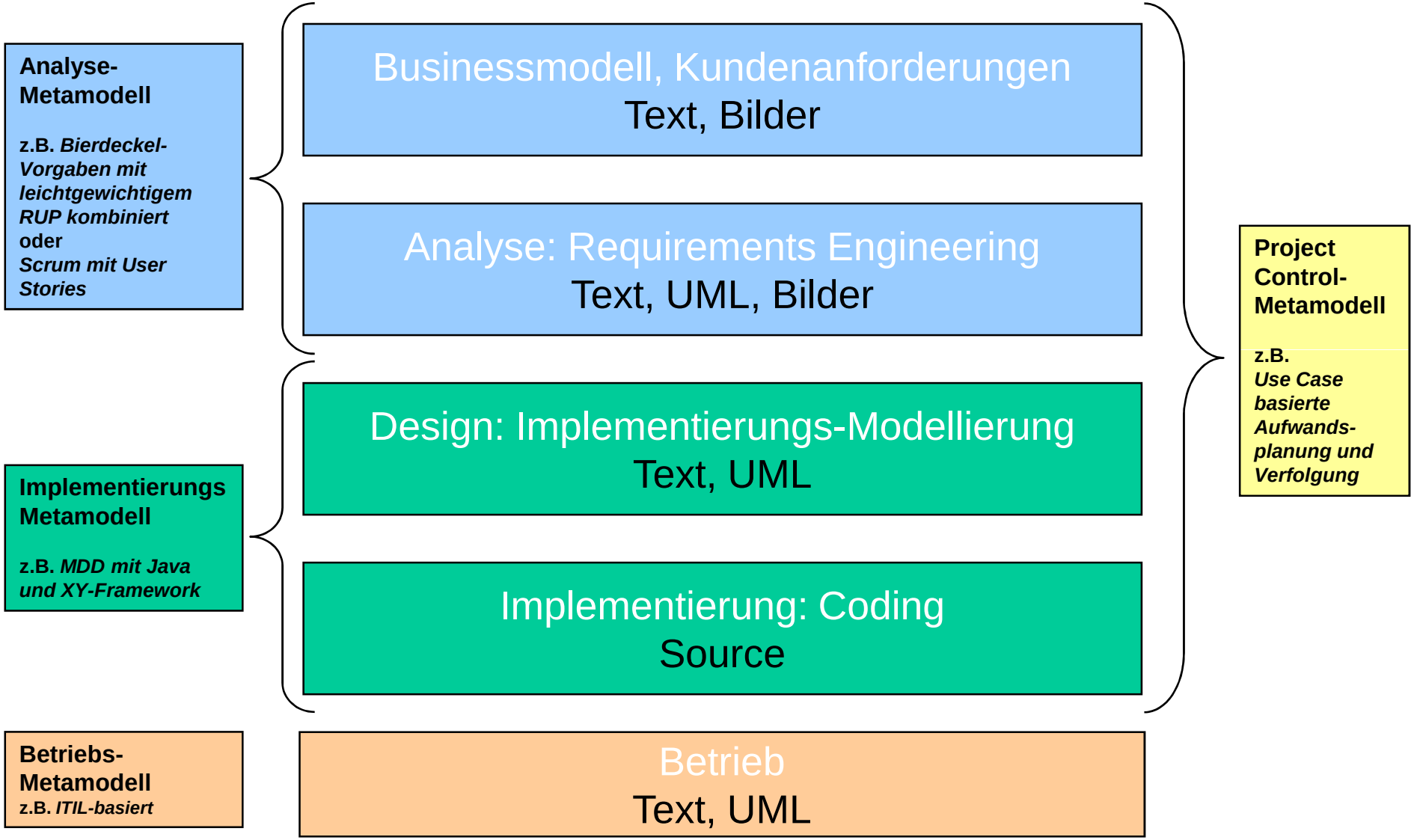
- **Lösung: Pflege einer Abbildung „Alt à Neu“**
  - z.B. modellbasiert auf Basis von Use Cases:
    Vertragsmodell **à** *Produktmodell*
  - sind die neuen Anforderungen umgesetzt, gelten die alten automatisch als erledigt

- **Schwierigkeiten:**
  - muss mit moderatem Aufwand leistbar sein, sonst macht es keiner; daher Toolsupport unerlässlich

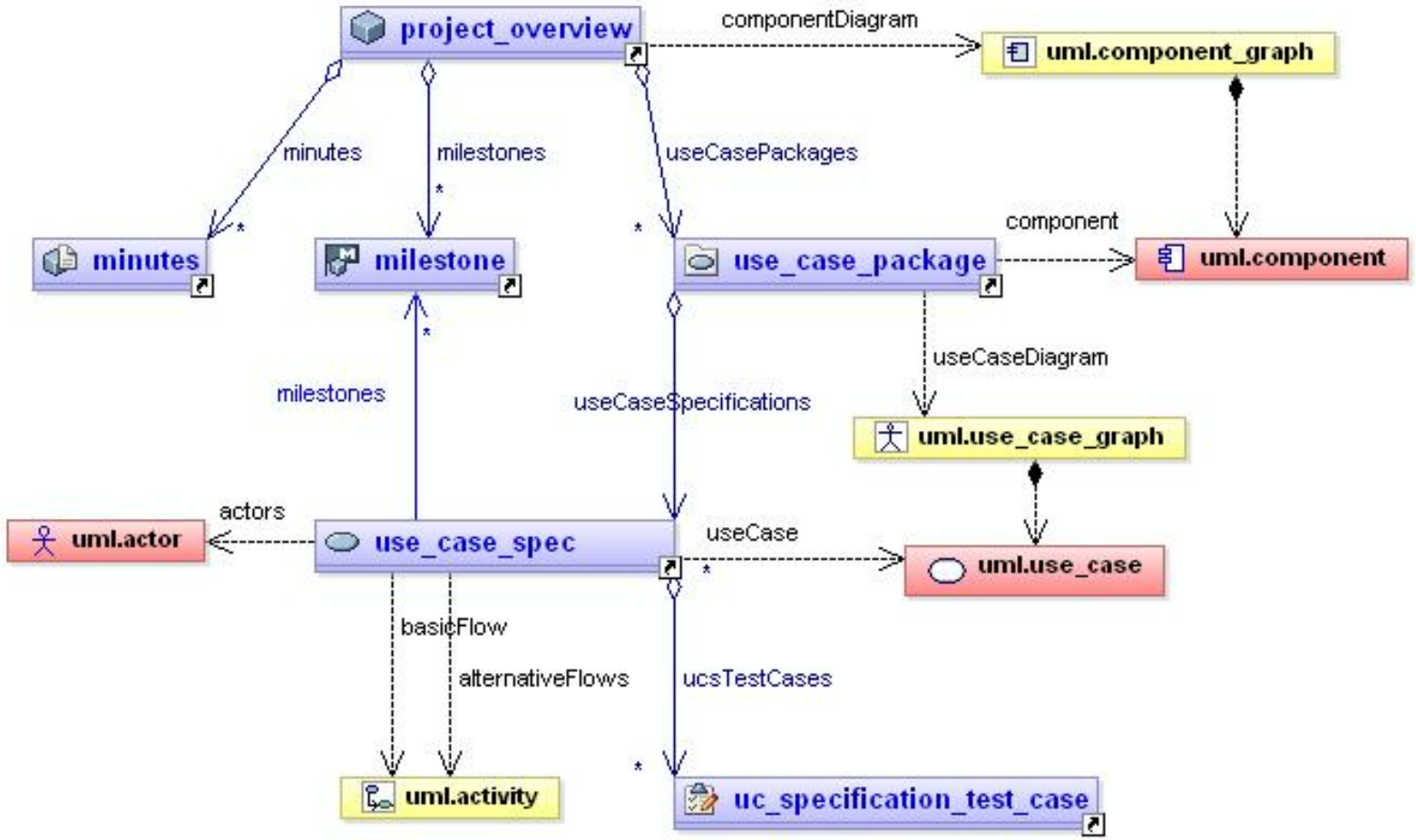
Agile Festpreisprojekte
Umgang mit Misstrauen

- **Problem: Kein Vertrauen des Auftraggebers in die Schätzungen des Auftragnehmers**
  - Umverteilen wird erschwert
  - „Das ist doch jetzt viel einfacher als das, was im Pflichtenheft steht“
- **Lösung: Transparenz schaffen à Vertrauen gewinnen**
  - ‚Gläserne Softwarefabrik‘
  - Entwickler des Kunden, falls vorhanden, mit einbeziehen
- **Schwierigkeiten**
  - Setzt hohen Reifegrad des Auftraggebers voraus
  - *... und wäre der voll gegeben, könnte man gleich ein 100% agiles Projekt machen...*

Anpassung des Vorgehens an den Projekttyp: Metamodelle nutzen



Anpassung des Vorgehens an den Projekttyp
Metamodell (Beispiel)



Vorteile des vorgestellten Ansatzes

- **Der Ansatz ist – relativ – leichtgewichtig**
  - Wenig methodische Regeln
  - Überschaubarer Aufwand für die Projektbeteiligten
- **Erweiterbar / adaptierbar durch Metamodell-Konzept**
  - Verschiedene Projekttypen von „klassisch“ bis „agil“ sind damit möglich
- **Anforderung an die Werkzeugunterstützung moderat**
  - Die benötigte Unterstützung lässt sich in die meisten UML Werkzeuge integrieren
- **Modellbasierte Entwicklungsprozesse arbeiten besonders effektiv mit dem Verfahren zusammen**

Beispielprojekte

- **Komplettes Warenwirtschaftssystem**
  - Umfang: groß
  - 350 grobe ⇒ 2500 feine Use Cases
  - Ca. 40 PJ Entwicklungsumfang
  - 560 DB Entitäten
- **Produkt für öffentlichen Verkehr (Aboverwaltung, etc...)**
  - Umfang: klein bis mittel (Migration einer existierenden Software)
  - 120 grobe ⇒ 240 feine Use Cases
  - Ca. 800 PT Entwicklungsumfang
  - 500 DB Entitäten
- **u.v.a.m.**

Ein Fazit aus dem praktischen Einsatz

- **Wir setzen das Vorgehen in diversen Projekten ein**
  - Doch jedes Projekt ist anders **à** Anpassung...
- **Das Vorgehen ist bei den Projektbeteiligten gut akzeptiert**
  - Steuerungsgremien
  - Kunden
  - Projektmitglieder
- **Der Ansatz wird „gelebt“**
- **Projekterfolge**
  - Es gab stets eine hohe Genauigkeit und Objektivität über den aktuellen Projektfortschritt
  - Projekttermine und Budgets konnten eingehalten werden