Projektorganisation

Christian Silberbauer

Motivation

- Benjamin Franklin:

> The bitterness of poor quality remains long after the sweetness of low price is forgotten.

Motivation

- Grady Booch, Object-Oriented Analysis and Design:

> Object-oriented software development requires that individual developers have unscheduled critical masses of time in which they can think, innovate, and develop, and meet informally with other team members as necessary to discuss detailed technical issues. The management team must plan for this unstructured time.

Einführung

- Kern der Projektorganisation ist die Realisierung von Projekten.
- Projekte haben ein definiertes Ziel.

- Träger von Projekten sind Organisationseinheiten.
- Gegenstände von Projekten sind Artefakte.
- Organisationseinheiten verarbeiten Artefakte im Rahmen von Projekten.

- Primäres Ziel des Projekts ist das Produkt.

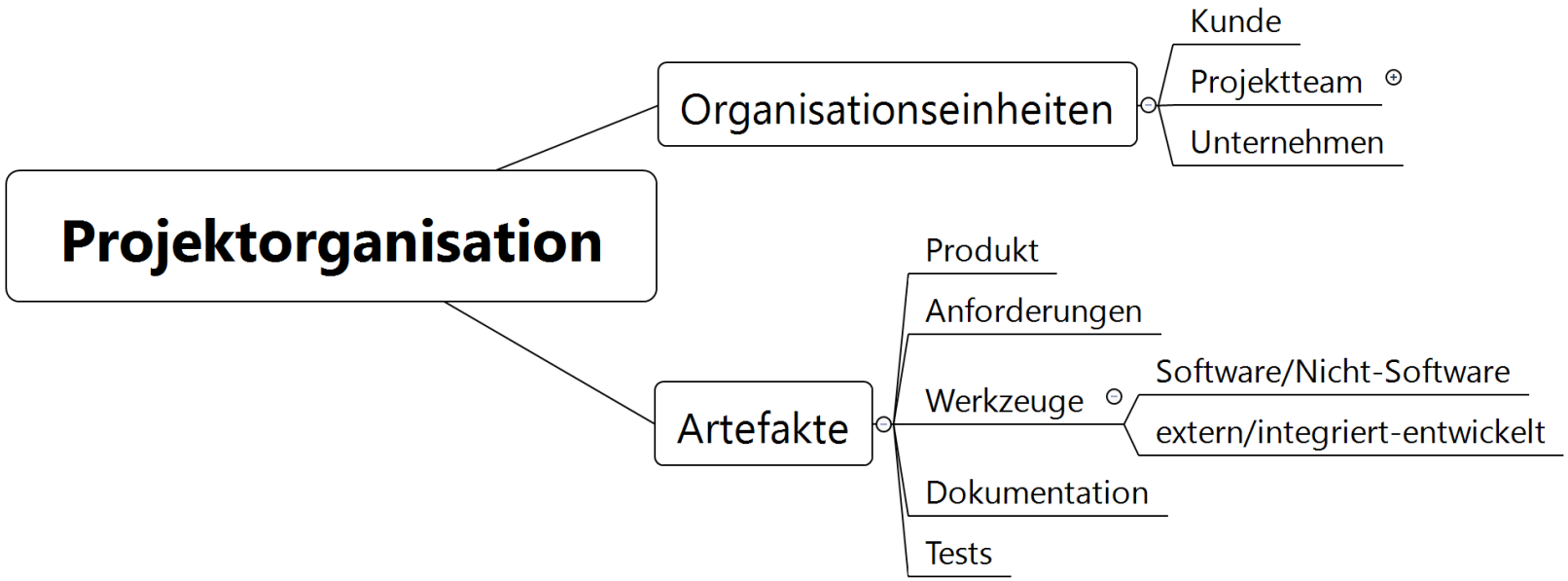
Organisationseinheiten und Artefakte

- Kern der Projektorganisation ist die Realisierung von Projekten.
- Projekte haben ein definiertes Ziel.

- Träger von Projekten sind Organisationseinheiten.
- Gegenstände von Projekten sind Artefakte.
- Organisationseinheiten verarbeiten Artefakte im Rahmen von Projekten.

- Primäres Ziel des Projekts ist das Produkt.

Organisationseinheiten und Artefakte



Organisationseinheiten

- Der Kunde stellt den Auftrag.
- Das Projektteam setzt diesen um.
- Das Unternehmen stellt dafür den Kontext dar und bietet insb. Projektressourcen.

Artefakte

- Im Wesentlichen wird das zu entwickelnde Produkt basierend auf den Anforderungen umgesetzt.

Werkzeuge

- Werkzeuge werden zur Entwicklung benutzt.
- In Softwareprojekten können dies Softwarewerkzeuge sein, z.B.:
  - Bibliotheken, Frameworks, generell wiederverwendbare Funktionen
  - Programmiersprachen
- Aber auch andere Werkzeuge, wie:
  - Hardware
  - Physische Umgebung

Werkzeuge

- Werkzeuge werden genutzt und sind auch sekundäres Ergebnis eines Projekts.
- Zur Wiederverwendung im Projekt und über das Projekt hinaus.

- Beispiele:
  - Integrierte Entwicklung eines Softwareframeworks
  - Code-Generatoren

Werkzeuge

- Werkzeugentwicklung bedeutet die Möglichkeit zur Effizienzsteigerung.
- Halbfertige Werkzeuge nützen wenig.
- Projektspezifische Anpassungen sollten daher nicht als bloße Workarounds verkommen, sondern sollten die Konsistenz des Werkzeugs wahren.

Dokumentation

- Dokumentation dient zur:
  - Projektumsetzung (Strategisch und Projektbezogen)
  - Nachvollziehbarkeit der Zusammenhänge (Traceability)
  - Verwendung des Produkts (User Manual)

- Letzteres geht in das Produkt ein.

Tests

- Tests dienen der Entwicklung zur Validierung des Produkts.
- Tests sind gegenüber den Anforderungen notwendig.
- Automatisierung der Tests trägt zur Effizienzsteigerung bei.

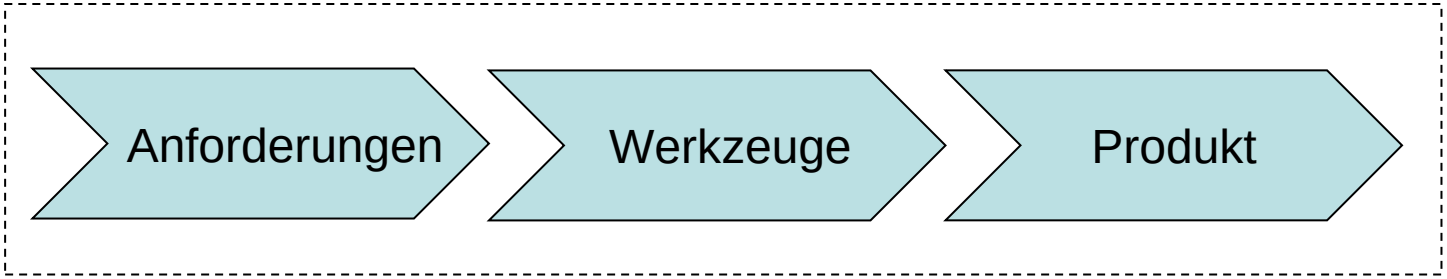
Werkzeuge++

- Die Stellung der Menschen in Projektorganisationen ist hervorgehoben.
- Komplexitätstheoretisch betrachtet, sind aber auch diese Werkzeuge, die zur Projektumsetzung dienen.

- Auch Tests stellen Werkzeuge dar.
- Nämlich zur Validierung des Produkts.

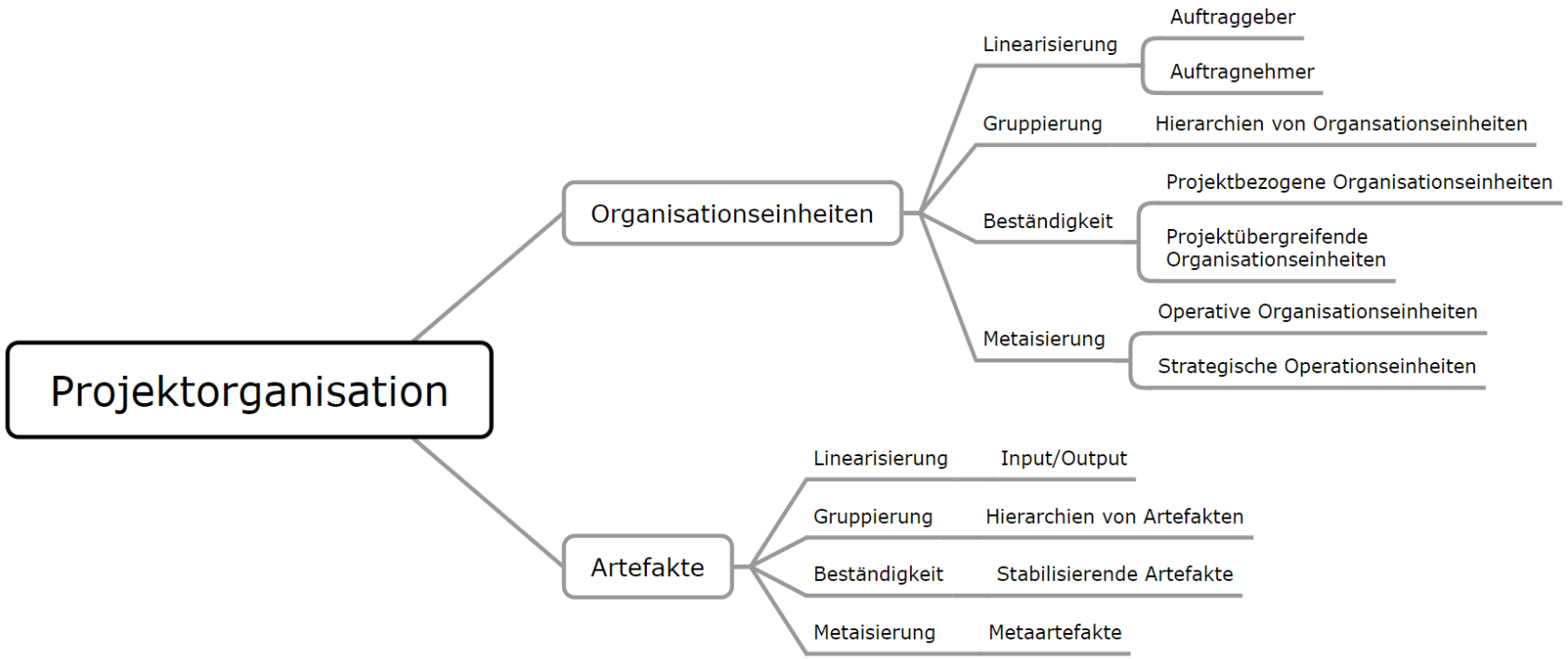
- Schließlich ist auch die Dokumentation Werkzeug für die Entwicklung.
- Ggf. abgesehen von der Benutzerdokumentation

Vereinfachte Darstellung



- Die erinnert an das EVA-Prinzip, dem Grundprinzip der Datenverarbeitung.

Aspekte


Projektorganisation
Organisationseinheiten
Linearisierung
Auftraggeber
Auftragnehmer
Gruppierung
Hierarchien von Organsationseinheiten
Beständigkeit
Projektbezogene Organisationseinheiten
Projektübergreifende Organisationseinheiten
Metaisierung
Operative Organisationseinheiten
Strategische Operationseinheiten
Artefakte
Linearisierung
Input/Output
Gruppierung
Hierarchien von Artefakten
Beständigkeit
Stabilisierende Artefakte
Metaisierung
Metaartefakte

Organisationseinheiten

- Linearisierung: Auftraggeber fordern bezüglich eines Projekts, Auftragnehmer sind gefordert. In letzter Instanz fordert der eigentliche Kunde, aber der Aspekt betrifft auch die unternehmensinterne „Befehlskette".
- Gruppierung: Organisationseinheiten bilden Hierarchien. Elementar sind einzelne Mitarbeiter. Sie gehören ggf. dem Projektteam an, sind aber auch ihrer jeweiligen Abteilung zugeordnet. Schließlich gehören sie zum Unternehmen.
- Beständigkeit: Integriert entwickelte Werkzeuge sorgen für Beständigkeit im Produkt und ggf. auch darüber hinaus. Sie sind stabilisierende Artefakte für das Projekt.
- Metaisierung: Vom eigentlichen lassen sich Metaartefakte abgrenzen. Z.B. Beschreibungen des Produkts wie Anforderungen und Dokumentation, Tests sowie Werkzeuge zur Entwicklung.

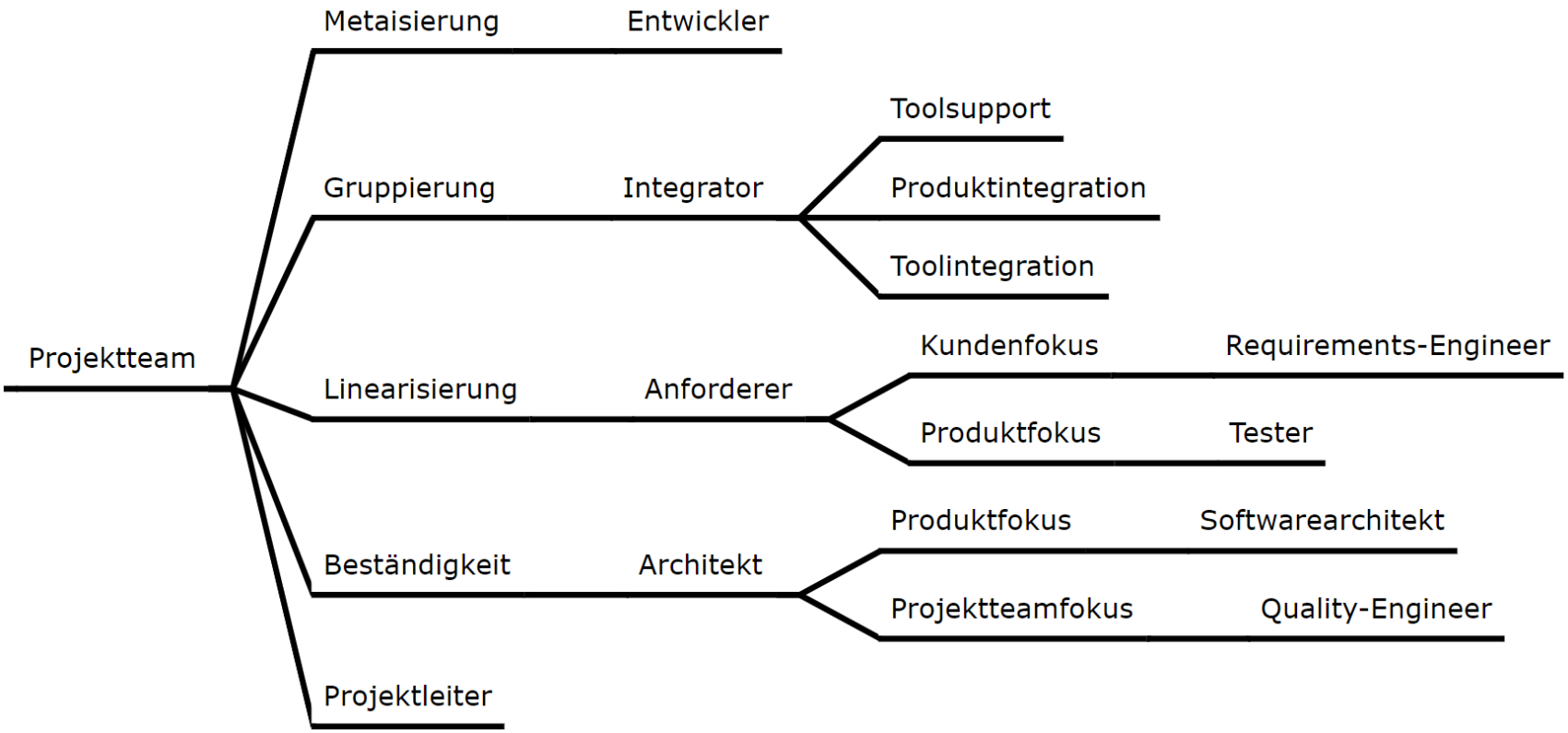
Artefakte

- Linearisierung: Input liefern Anforderungen an das Projekt oder gegebene Werkzeuge. Output ist primär das eigentliche Produkt, Tests und wiederum sonstige Werkzeuge.
- Gruppierung: Das Produkt lässt sich in Teilprodukte unterteilen, was wiederum eine verteilte Entwicklung erlaubt. Die Bestandteile des Produkts bilden so eine Hierarchie. Auch der Kontext in Entwicklung und Verteilung von Elementen ist zu berücksichtigen.
- Beständigkeit: Projektbezogene und projektübergreifende Organisationseinheiten sind zu unterscheiden. Projektübergreifende Organisationseinheiten überdauern die Projektlaufzeit.
- Metaisierung: Der Fokus von OEs kann eher operativ oder strategisch sein, also eher das „wie" oder eher das „was" bestimmend.

Rollen

- Softwareentwicklungsprojekte sind komplex.
- Ein Projekt kann einfacher bewältigt werden, wenn die Rollenverteilung optimal ist.
- Förderlich ist dabei der Fokus auf die grundlegenden Differenzierungsaspekte.

Rollen


Projektteam
Metaisierung
Entwickler
Gruppierung
Integrator
Toolsupport
Produktintegration
Toolintegration
Linearisierung
Anforderer
Kundenfokus
Requirements-Engineer
Produktfokus
Tester
Beständigkeit
Architekt
Produktfokus
Softwarearchitekt
Projektteamfokus
Quality-Engineer
Projektleiter

Rollen

- Metaisierung: Ein Entwickler erhält Anforderungen und setzt diese in Form eines Programms um. Er bekommt als Input also „was“ programmiert werden soll und bestimmt dann „wie“.
- Gruppierung: Ein Integrator führt im Projekt die Teile des Produkts zusammen und auch integriert entwickelten Werkzeuge. Zudem stellt er Entwicklungswerkzeuge bereit.
- Linearisierung: Ein „Anforderer ist spezialisiert auf die Weitergabe von Informationen an den Entwickler und vom Entwickler. Als Requirements-Engineer klärt und filtert und aufbereitet er Anforderungen mit dem Kunden für den Entwickler. Zudem nimmt er das Feedback des Entwicklers auf und kommuniziert es adäquat mit dem Kunden. Als Tester prüft er möglichst unabhängig von der Entwicklung das entwickelte Produkt in seinen Teilen und auch im Ganzen.

Rollen

- Beständigkeit: Der Architekt schafft Standards für die Entwicklung. Er definiert, was stabil bleiben muss und wo demgegenüber Freiräume übrigbleiben. Der Architekt bestimmt Strukturen des Produkts und des Prozesses, schränkt damit die Flexibilität des Entwicklers ein und reduziert so die Komplexität der Entwicklung. Das schaffen von Strukturen hat das Potenzial für wiederverwendbare Teilprodukte, also von Werkzeugen. Der Architekt führt die integrierte Entwicklung von Werkzeugen. Dies umfasst Softwarewerkzeuge genauso wie die Definition und Umsetzung des Entwicklungsprozesses. Er setzt z.B. die Rahmenbedingungen für Programmierrichtlinien, Code-Reviews, Software-Tests, die Dokumentation (Templates), das Versions- und Änderungs-Management. Üblicherweise wird die Rolle in den eigentlichen Softwarearchitekten und einem Quality-Engineer zerlegt. Ersterer deckt technische Aspekte zur eigentlichen Entwicklung ab, letzterer hat vielmehr das Projektteam im Blick, also Zusammenspiel der Teammitglieder.

Rollen

- Der Projektleiter schließlich koordiniert die verschiedenen Rollen im Projekt, repräsentiert das Projekt gegenüber dem Kunden und verantwortet wirtschaftliche Aspekte. Eine der vier grundlegenden Differenzierungen kann ihm nicht zugeordnet werden. Stattdessen schafft er die Verbindung der Projektteile. Sein Fokus ist das Projekt als Einheit und die Verbindung einzelner Bestandteile und Aufgaben (Traceability). Seine Rolle hat die geringste Spezialisierung und ist am stärksten geprägt von Heterogenität.

Rollen

- Der Fokus von Rollen auf einzelne Differenzierungsaspekte ist nicht exklusiv.
- Ein „Über den Tellerrand“ hinaus blicken ist notwendig, um Lücken zu schließen.
- Einzelne Rollen können natürlich weiter zerlegt und auch personell zusammengefasst werden.

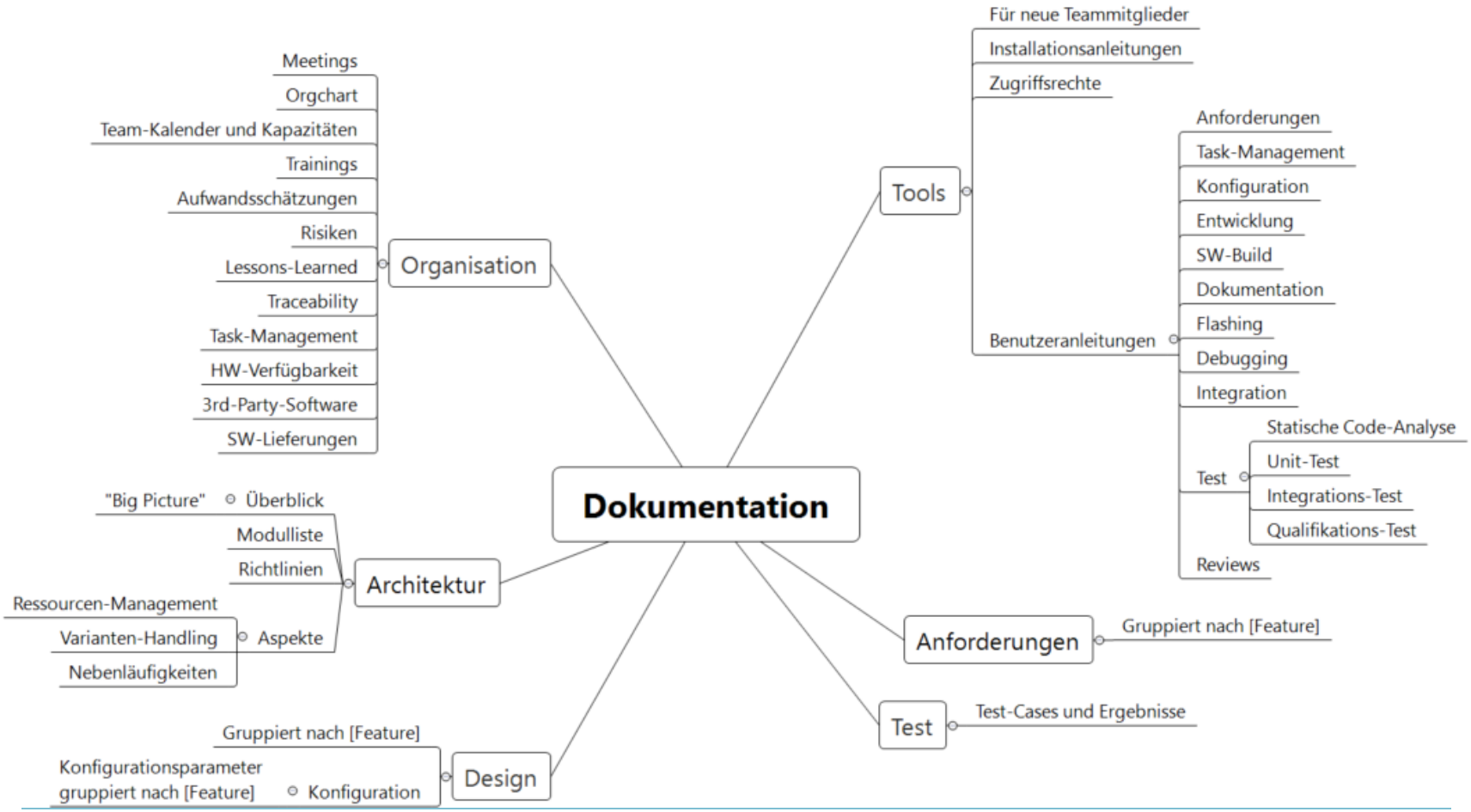
- Häufige Kontextwechsel sind kontraproduktiv („Rüstkosten“).

Dokumentation

- Dokumentationen werden gepflegt extern gegenüber dem Kunden und intern innerhalb des Projekts und projektübergreifend.

- Die Struktur für eine projektinterne Dokumentation für eine embedded Software sieht beispielhalft wie folgt aus:

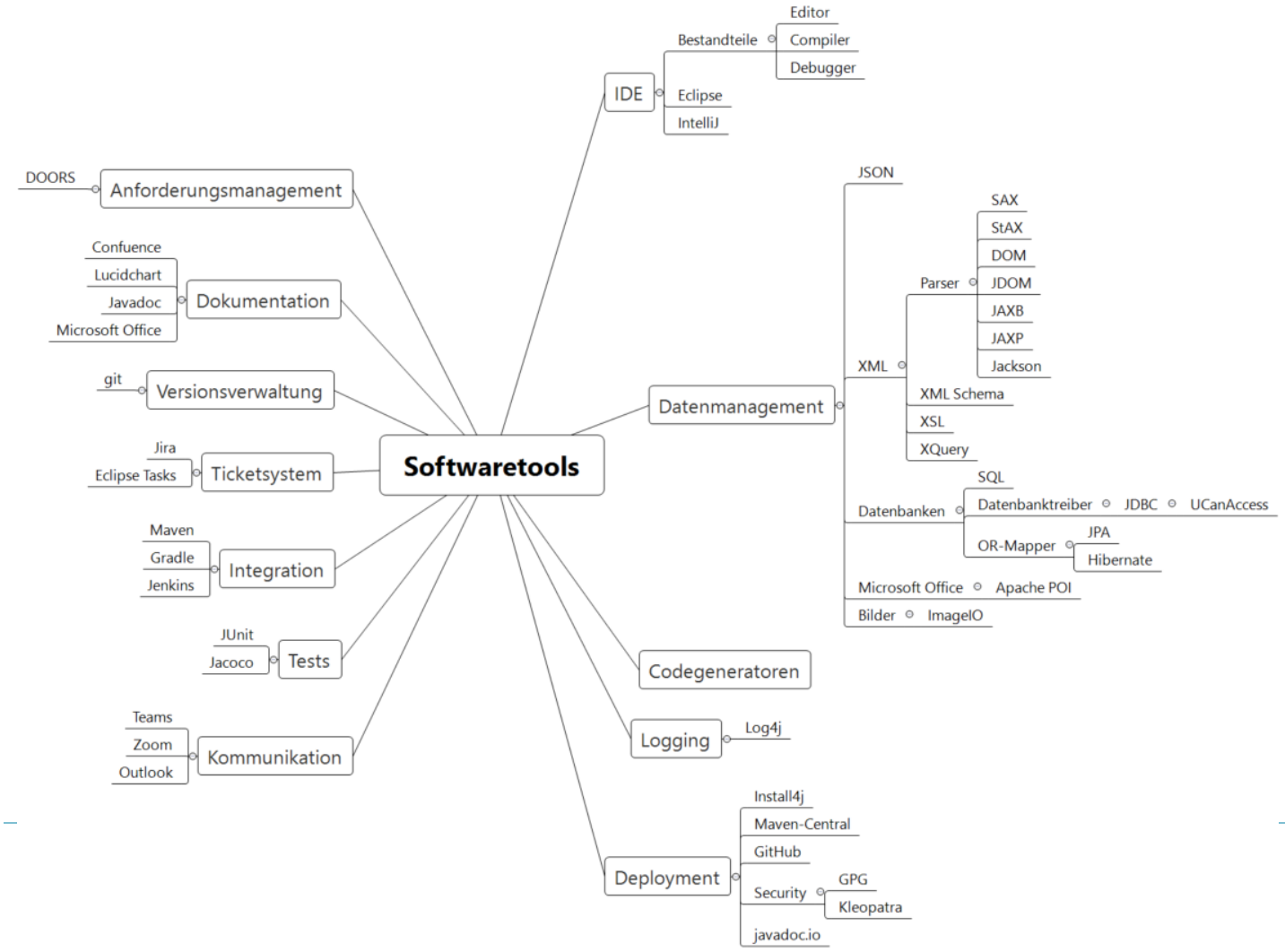
Dokumentation



Softwaretools

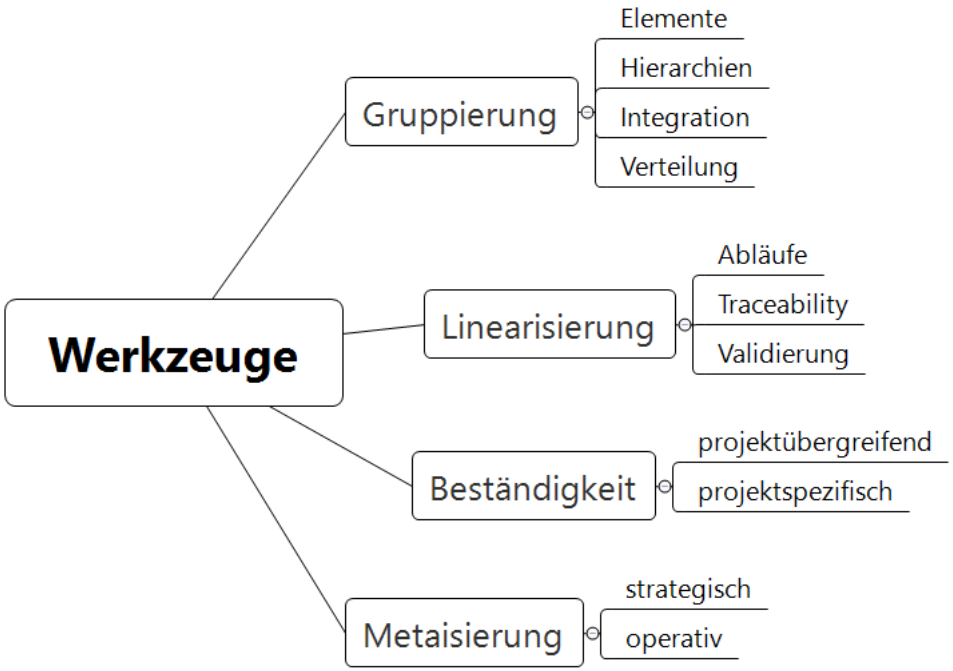
- Einen beispielhaften Überblick über Softwaretools in einem Javaprojekt gibt folgende Folie.

Softwaretools



Aspekte

- Die Vollständigkeit von Werkzeugen wie z.B. Dokumentation und Softwaretools lässt sich anhand folgenden Schemas erarbeiten:



Integrierte Werkzeugentwicklung

- Integrierte Werkzeugentwicklung bedeutet einen Beitrag zur Effizienzsteigerung im Projekt und darüber hinaus zu leisten.
- Sie trägt zur Stabilisierung des Unternehmens bei und ist damit mit hoher Priorität zu berücksichtigen.
- Der Architekt verantwortet diese innerhalb des Projekts.
- Eine Einbindung des Architekts in übergreifende Organisationseinheiten ist für derartige Bestrebungen förderlich.

Projektübergreifend?

- Projekte werden oftmals als abgeschlossene Einheiten betrachtet.
- Fließen ihre Tools, Erkenntnisse aber nicht in übergreifend Organisationseinheiten ein, bleibt übergreifend nur eine Personenabhängigkeit.
- Das Potenzial kontinuierlicher Weiterentwicklung von Projekt zu Projekt wird so wenig genutzt.

- Es ist kostspielig, das Rad immer wieder neu zu erfinden…

Refactoring

- Es ist unrealistisch in komplexen Projekten von Anfang an die einzelnen Schritte präzise planen zu können.
- Im Laufe eines Projekts entwickelt sich auch die Erkenntnis über Strukturen.
- Dies erfordert Refactoring bzw. Restrukturierung.
- Ein Aufwand, der keinen funktionalen Fortschritt herbeiführt, aber einer der ein Projekt vor „Vermüllung“ schützt.

- Transparenz für Refactoring ist notwendig.

Feedback

- Eine hierarchische „Befehlskette“ von oben nach unten ist charmant.
- Ohne Feedback („Bottom-Up-Kommunikation) funktioniert sie aber nur bei trivialen Softwareprojekten oder bei einem exzellent eingespielten Team.

- In aller Regel ist Feedback notwendig.

- Möglichkeiten:
  - Regelmeetings
  - Dailys
  - Lessons-Learned
  - Informelle Treffen („Teeküchengespräch“)

Feedback

- Weniger explizites Feedback ist durch ein harmonisches Team erreichbar.

- Z.B. durch:
  - Gute Teamzusammenstellung
  - Kick-Off-Workshops
  - Team-Events

Traceability

- Es ist notwendig zu wissen, wo man in einem Projekt steht.
- Dafür benötigt es eine Zeit- und Aufgabenplanung.
- Aufgaben werden in Teilaufgaben zerlegt.
- Tests/Validierungen werden auf Teilaspekte angewandt.
- Nachvollziehbarkeit und Konsistenz ist sicherzustellen.
- Problemfelder:
  - Mehrdimensionalität von Projekten (Werkzeugentwicklung)
  - Notwendigkeit von Refactoring

- Softwareprojekte sind zu komplex, um sie vollständig vorab planen zu können.
- Projektplanung ist daher während des gesamten Projektverlaufs notwendig.

Traceability

- Mehr Traceability ist möglich durch:
  - Feingranularere Aufgabenplanung
  - Mehr Dokumentation
  - Häufigere (System-)Integration
  - Mehr Testaufwand
  - Mehr Abstimmungsmeetings
- Aber: der Mehraufwand dafür muss sich rechnen!

Automatisierung

- An Automatisierung in Projekt zu arbeiten, führt zur Effizienzsteigerung.
- Sie ist daher adäquat hoch zu priorisieren.

- Projektübergreifender Toolsupport ist dabei hilfreich.

Literatur

- Booch, Grady. 1993. *Object-Oriented Analysis and Design, 2nd Edition*. Addison-Wesley