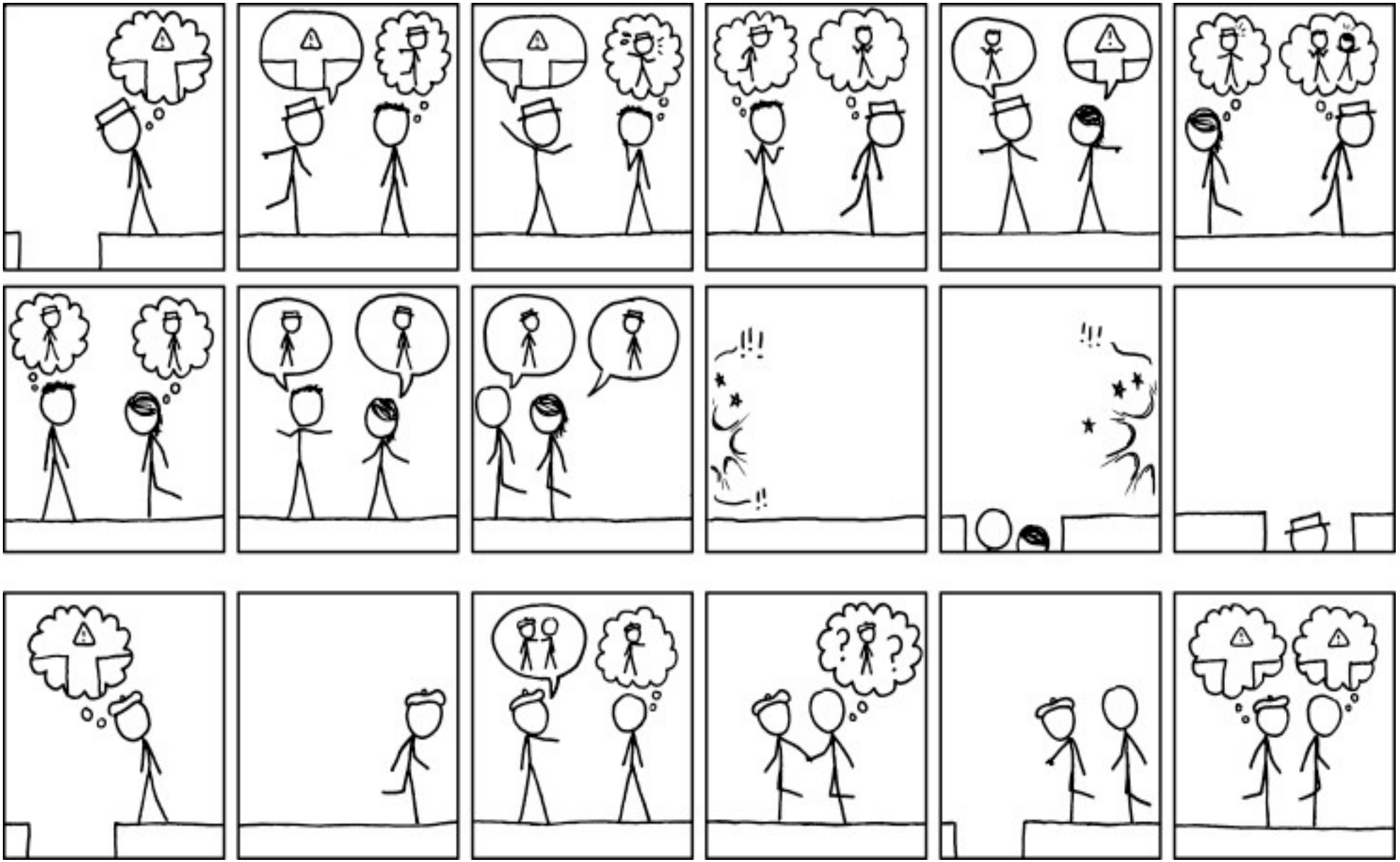
Softwareentwicklungsmodelle



Gliederung

- Der Softwareentwicklungsprozess
- Wasserfallmodell
- V-Modell
- Prototypen-Modell
- Spiralmodell
- Agile

Warum

- Fall 1: Ich entwickle zu meinem persönlichen Spaß ein Schachprogramm als Freeware
- Fall 2: Eine große Firma entwickelt ein Schachprogramm, das den Weltmeister schlagen soll
- Fall 3: Leute die nichts von Schach verstehen entwickeln ein Schachprogramm, das zumindest ein Konkurrenzunternehmen schlagen soll

Persönlicher Softwareprozess

- Diverse Fragen die ich mir selbst beantworten muß
  - für welche Zielumgebung schreibe ich mein Programm?
  - Wie soll es an der Oberfläche aussehen
  - Wie sieht meine grobe „Architektur“ aus
  - Welche Features möchte ich bieten
    - Das Programm soll gegen mich spielen können
    - es Partien analysieren könne
    - es soll auch Schachtrainer sein
  - Etc..

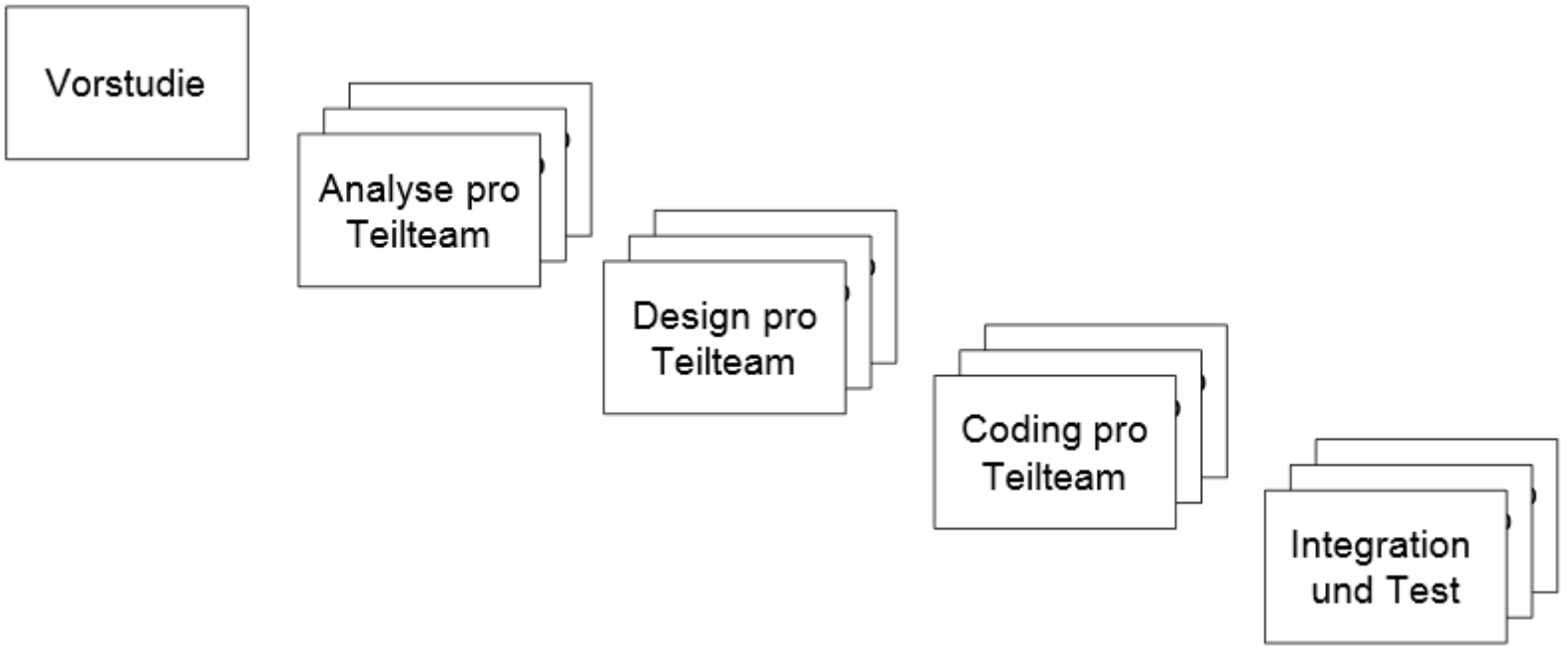
Persönlicher Softwareprozess

- Fragen, die ich mir oft nicht beantworte
- wie lange werde ich brauchen?
- was wird es kosten
- welche Spielstärke wird das Programm in Elo-Punkten haben
- soll ich eine formale Spezifikation verwenden oder nicht

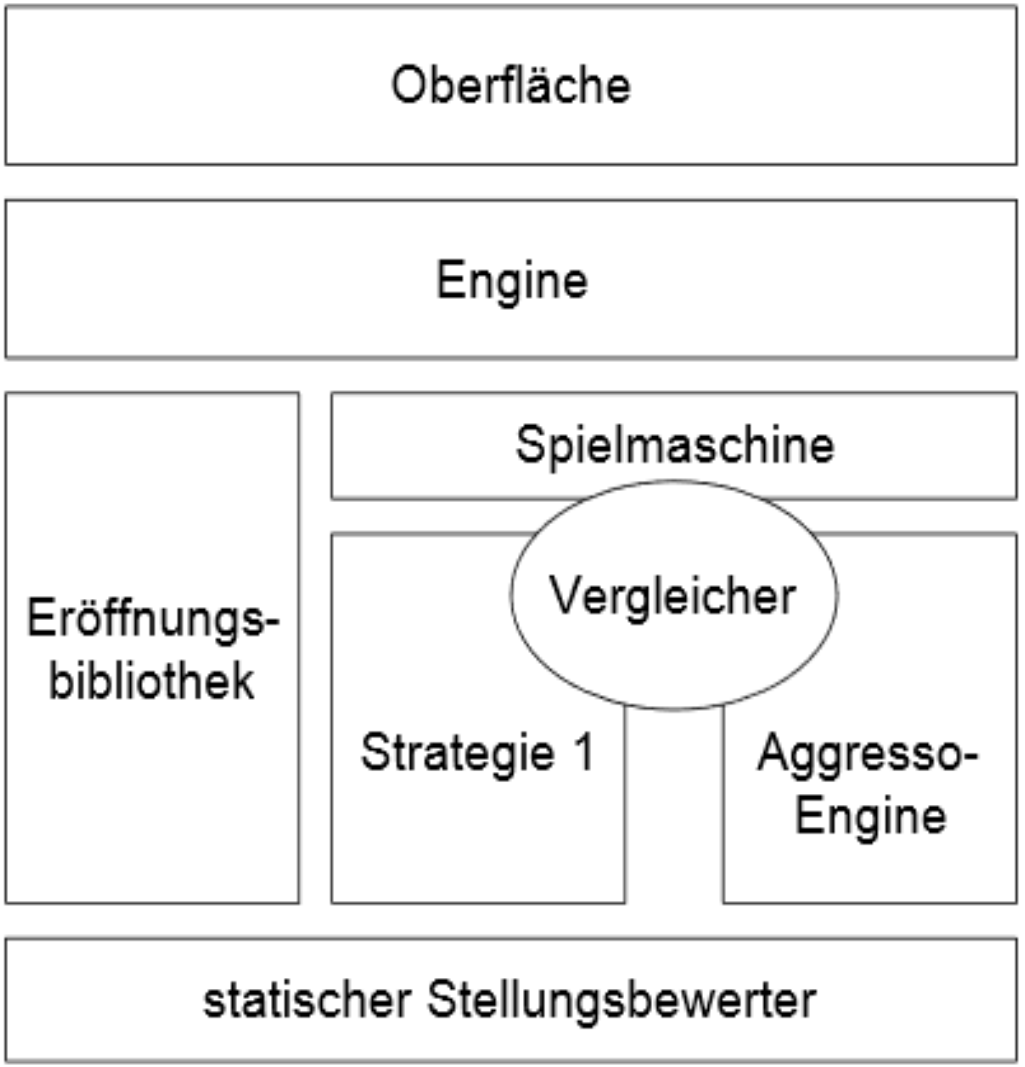
Die große Firma möchte den Weltmeister schlagen

- Es wird zunächst eine Machbarkeitsstudie erstellt werden, anhand derer das Problem vorstrukturiert wird
- Dann wird eine grobe Architektur erstellt, anhand derer auch die Teilteams aufgeteilt werden
- Jedes Teilteam analysiert seinen Bereich, designed ihn und implementiert und testet ihn
- am Ende passt hoffentlich alles zusammen

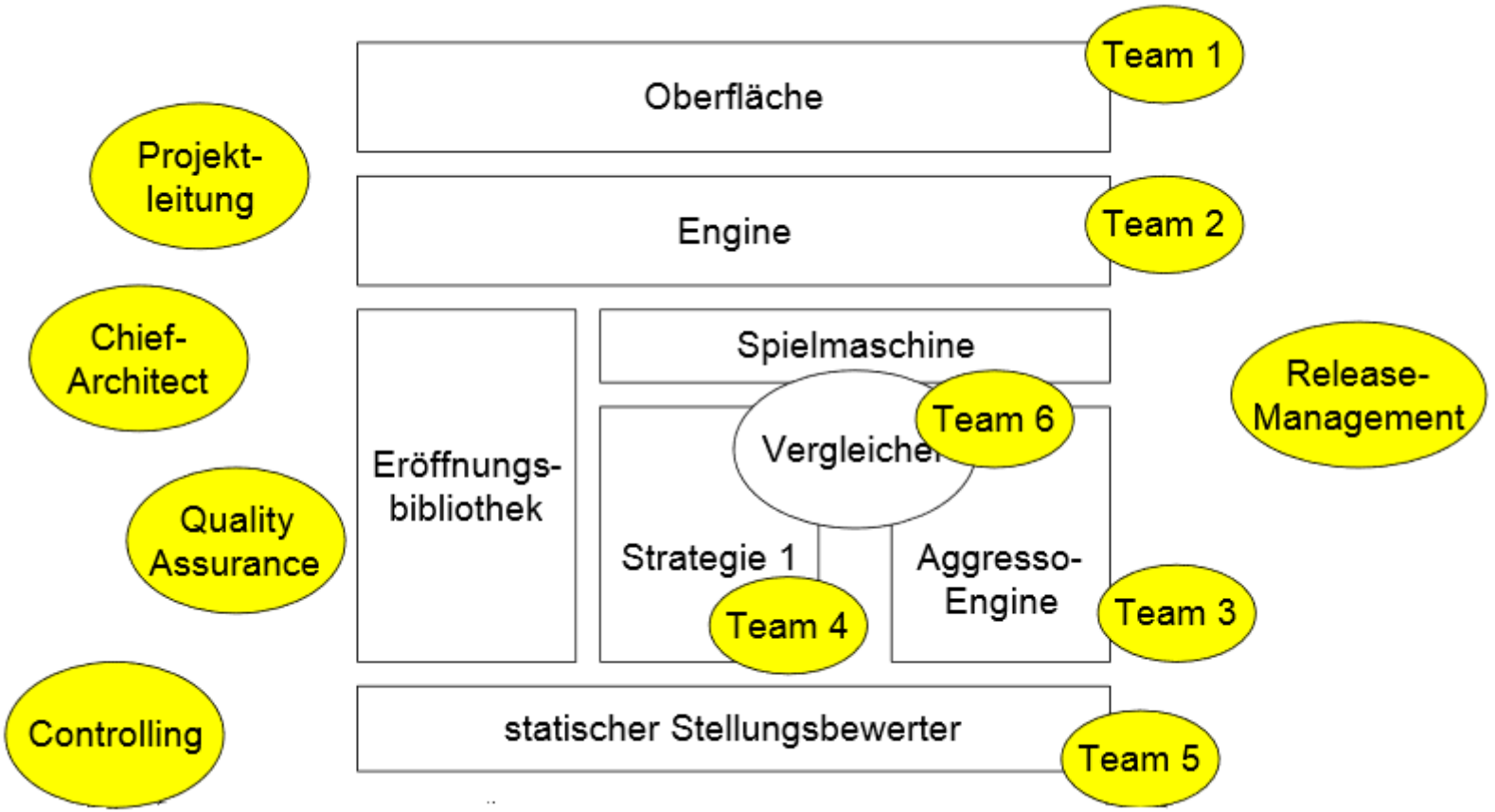
Deep Chess Architektur



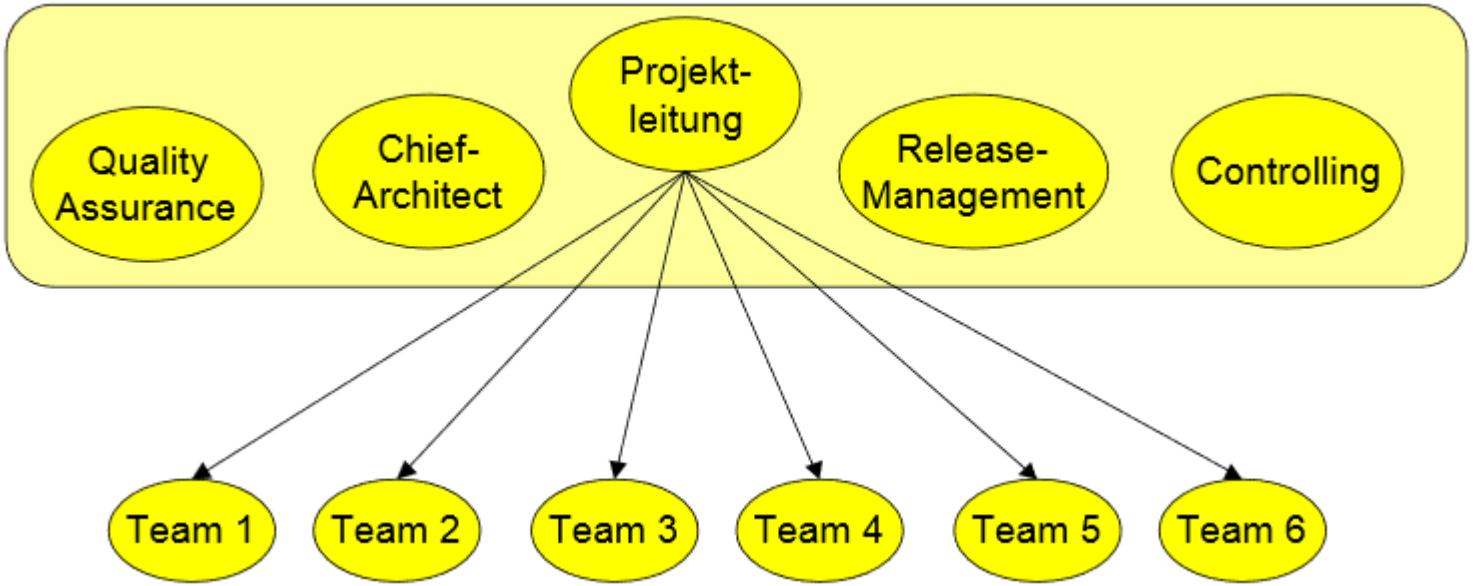
Deep Chess Architektur



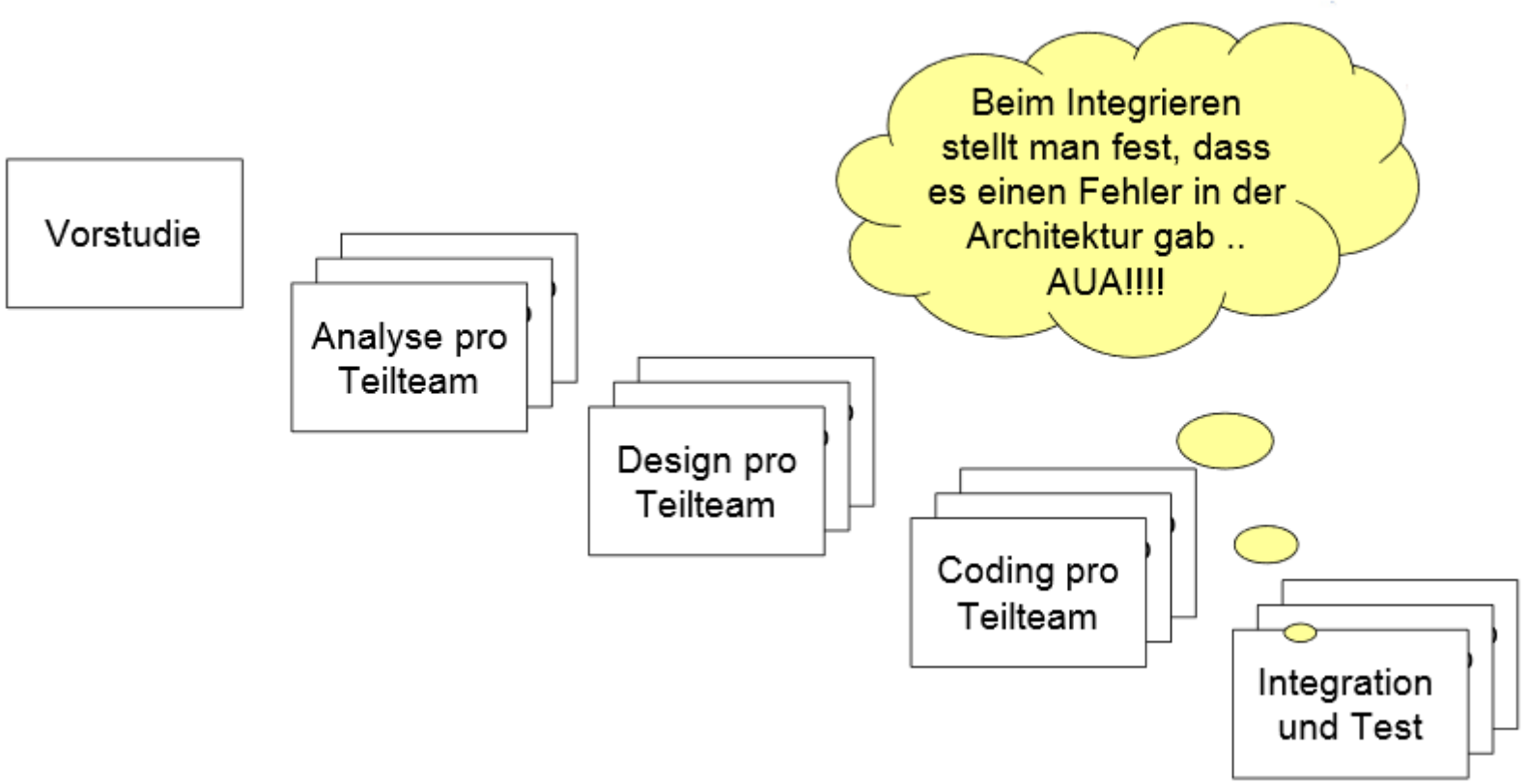
Deep Chess Architektur



Deep Chess Architektur



Deep Chess Architektur



Ein paar Einflußfaktoren also

- Der optimale Prozess hat etwas damit zu tun, wieviele Leute entwickeln
- Der Prozess ist damit beeinflusst von der Teamgröße
- Die Teamgröße wird von den Requirements und der Deadline beeinflußt
- Die Teamgröße wird von der Projektgröße beeinflusst
- Die Architektur ist auch Folge eines sozialen Prozesses

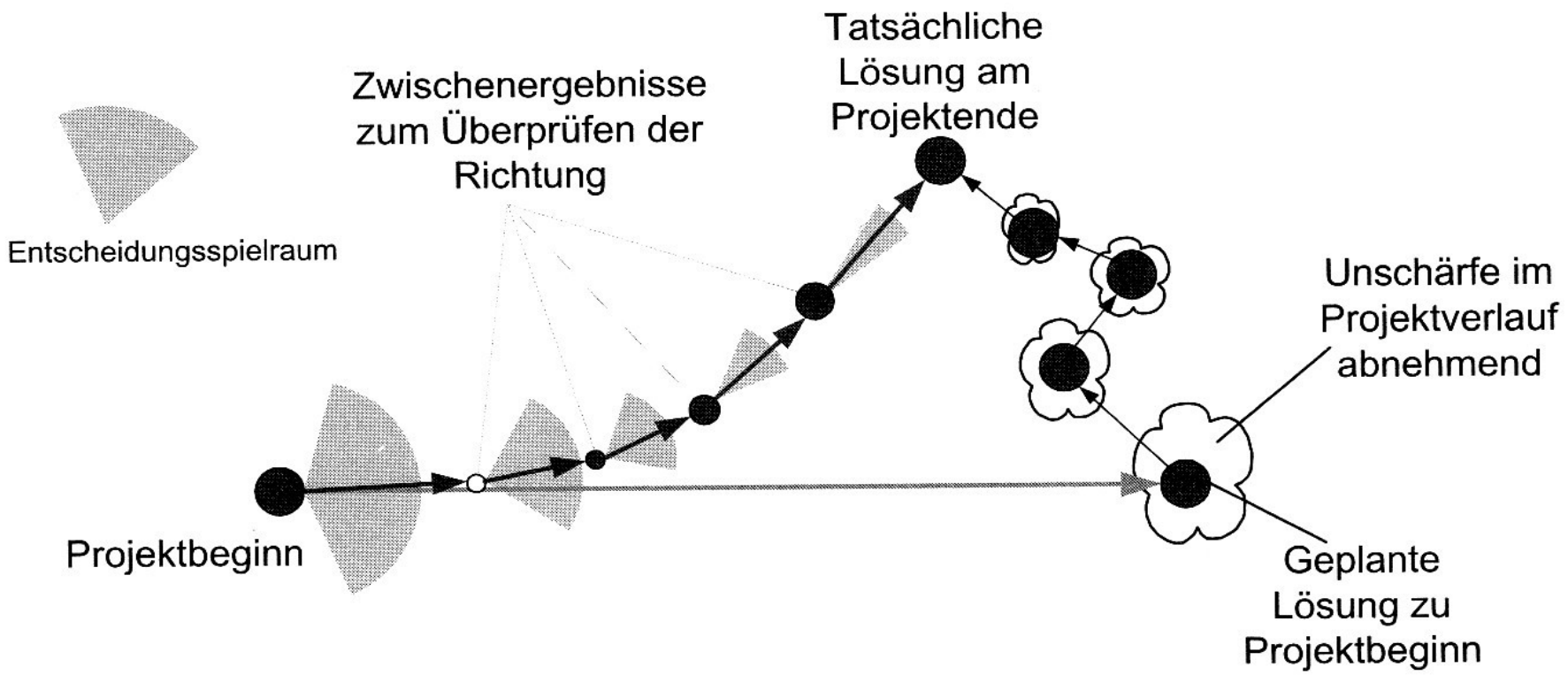
10 Leute, die von Schach nichts verstehen, bauen ein Schachprogramm

- Sie werden die Fachabteilung fragen und mit der eine Vorstudie und Analyse machen
- Alles wird länger dauern, als in Fall 2
- Es wird mehr Fehler geben
- Das Ergebnis wird schlechter

Frage dann noch

- Wie viel Sinn macht ein absolut einheitlicher Prozess für jede Art von Projekt in einer Firma?
- Es gibt mehr als eine Firma, die diesem Ideal hinterherläuft?
- Begriff dazu (sinnvoller) Tayloring: Der Prozess wird an das Projekt angepasst, bevor es losgeht

Problem der Unsicherheit und Risiken



Der Softwareentwicklungsprozess

- Organisation der Softwareentwicklung in beherrschbare Schritte
- Arbeitsschritte
  - Analyse
  - Design
  - Implementierung
  - Testen

Analyse

- Was soll das System können?
- Anforderungsanalyse
  - Analyse der Kundenwünsche
- Analyse der Machbarkeit
  - Risikoanalyse
  - Kosten

Entwurf

- Wie soll das System funktionieren?
- oft gegliedert in Grobentwurf/Feinentwurf
- Übergang zwischen Analyse und Design oft fließend

System Design

- Architektur des Systems
  - Strukturierung in Komponenten
  - Package etc
  - Festlegung von Schnittstellen
- Einarbeitung technischer Randbedingungen
  - Was soll in die Datenbank?
  - Was in welcher Programmiersprache?
  - Welche Bibliotheken verwenden?

Program Design

konkrete Festlegung von

- Datentypen
- Einzelklassen
- Algorithmen
- Ein/Ausgabe von Funktionen (Methoden)
- Abgleich mit ermitteltem Aufwand
- Aufgabenverteilung an Programmierer

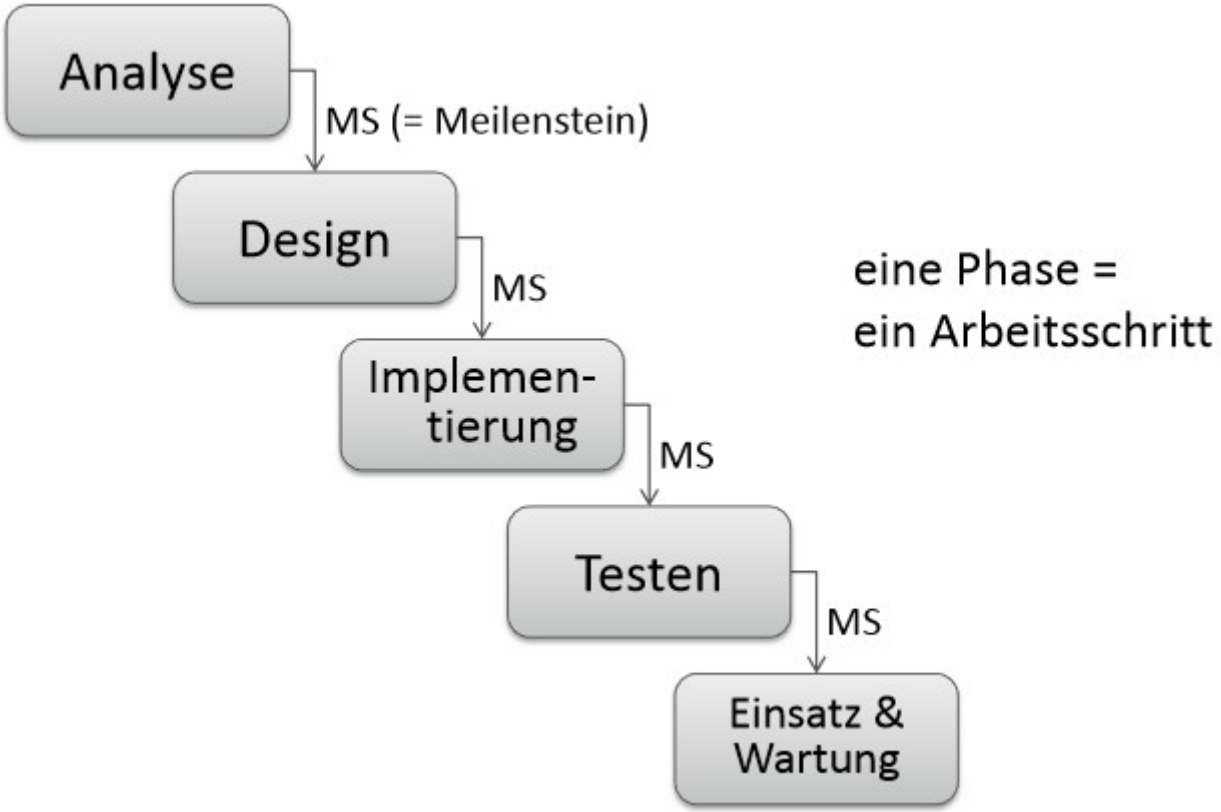
Implementierung

- Umsetzung des Designs in Code
- Programmier und Dokumentationsrichtlinien
- Versions und Konfigurationsmanagement
- Entwicklergruppen etc

Testen

- Erfüllt das System die Anforderungen?
- Überprüfen, ob System korrekt auf Eingaben reagiert
- Unterscheidung zwischen
  - Komponententest
  - Integrationstest
  - Systemtest

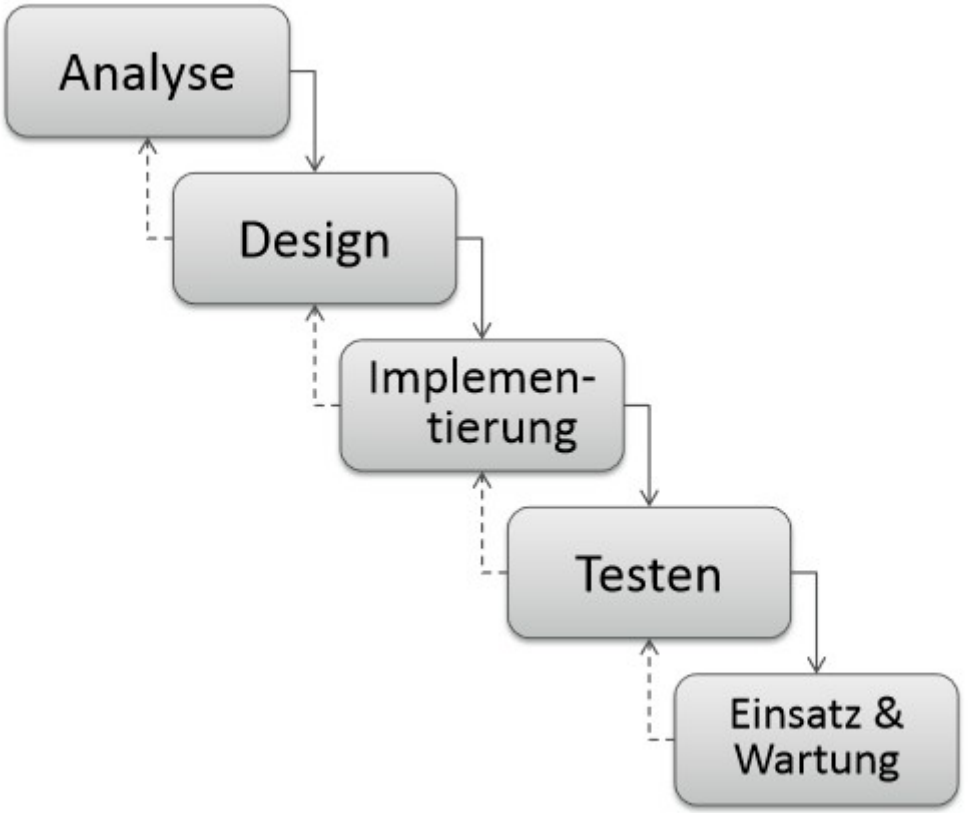
Wasserfallmodell



Wasserfallmodell

- idealisierter sequentieller Ablauf
- nicht auf Korrekturen ausgelegt
- Was tun bei Problemen
  - Design zeigt, dass Analyse unvollständig
  - Implementierung zeigt, dass Datenstrukturen/Schnittstellen fehlen
  - Fehler beim Testen
- Erweiterung: iteriertes Wasserfallmodell

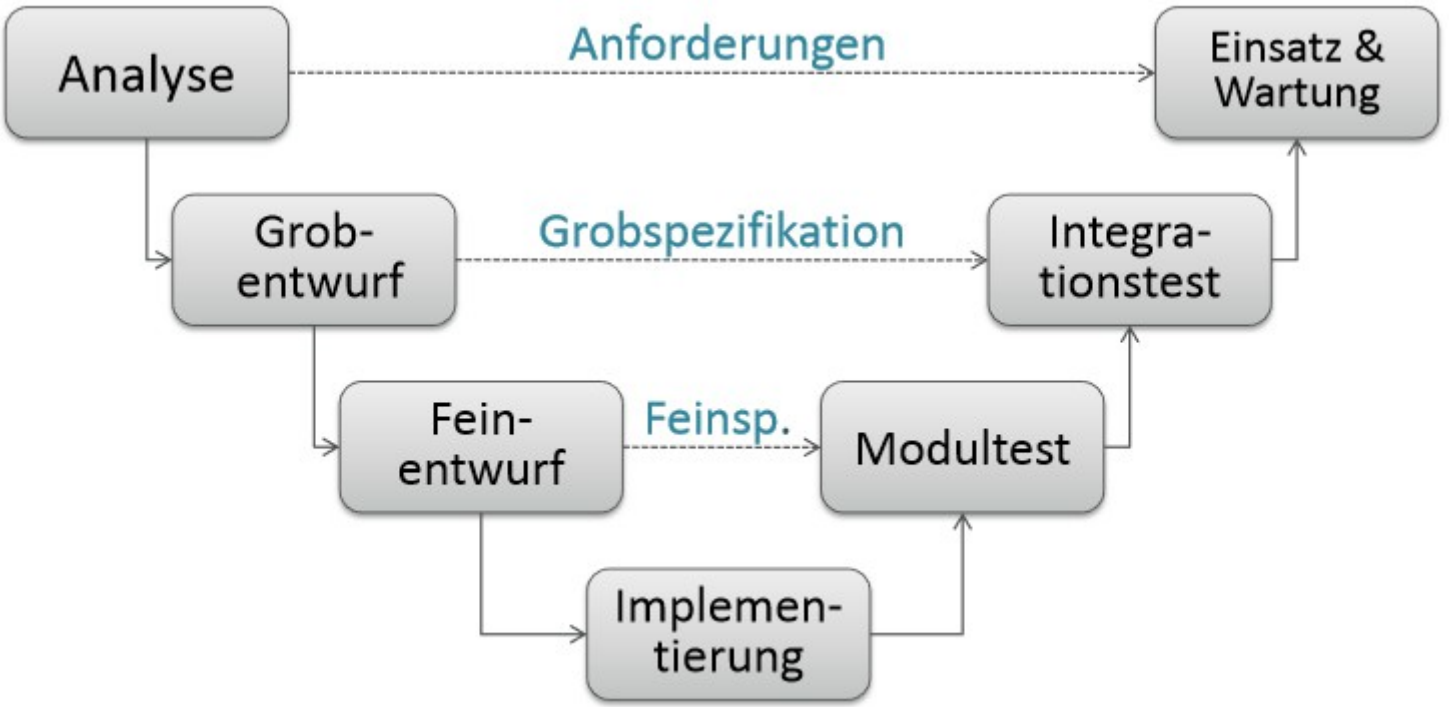
Iteriertes Wasserfallmodell



Iteriertes Wasserfallmodell

- „Testen“ (Analyse des erzeugten Codes) ist nicht gekoppelt mit „Entwicklung“ (Synthese)
- Entwicklung sollte Dokumente ergeben, die für das Testen geeignet sind
- Resultat: V-Modell

V-Modell


Analyse
Anforderungen
Einsatz & Wartung
Grob-entwurf
Grobspezifikation
Integra-tionstest
Fein-entwurf
Feinsp.
Modultest
Implemen-tierung

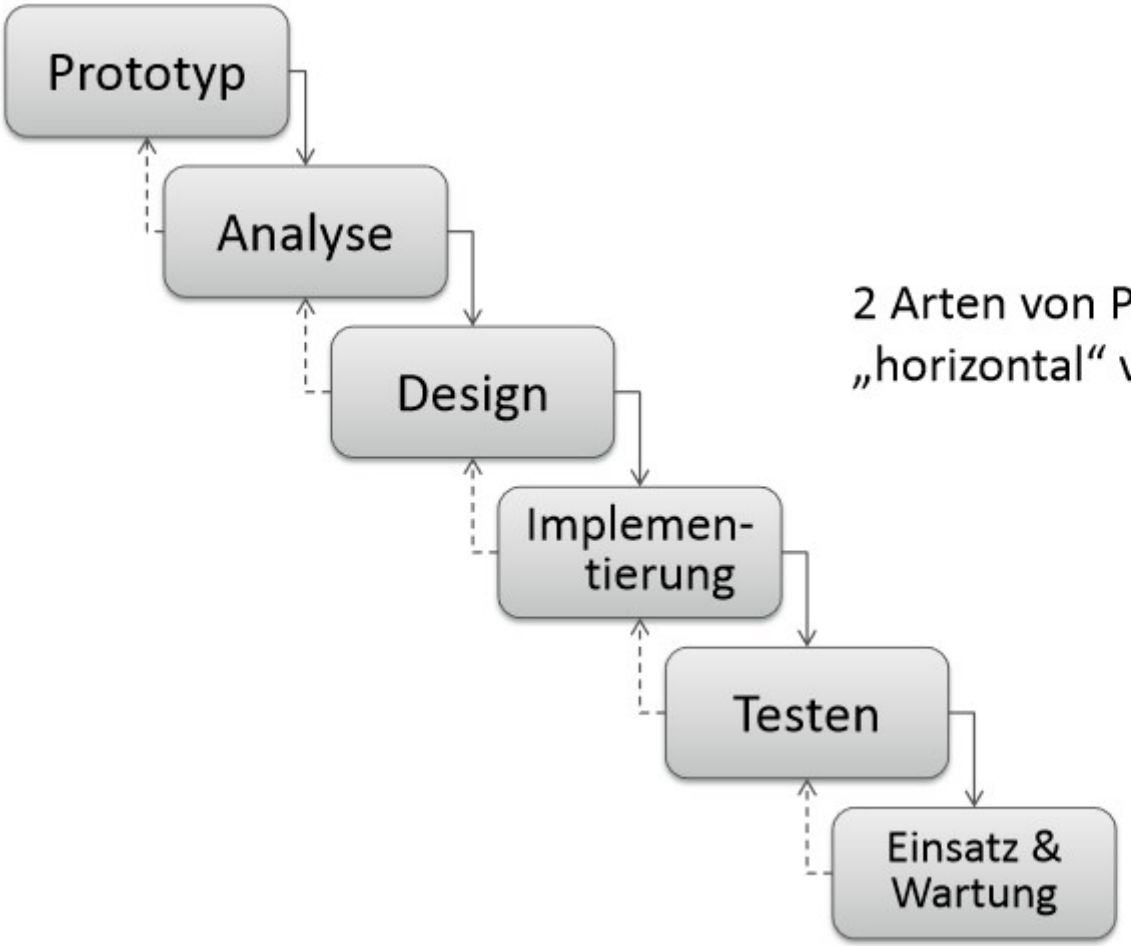
V-Modell

- Anwender(an)forderungen
- Technische Anforderungen
- Algorithmendesign
- Implementierung
- Implementierungsdokumentation
- Prüfprozedur, Prüfergebnis

V-Modell

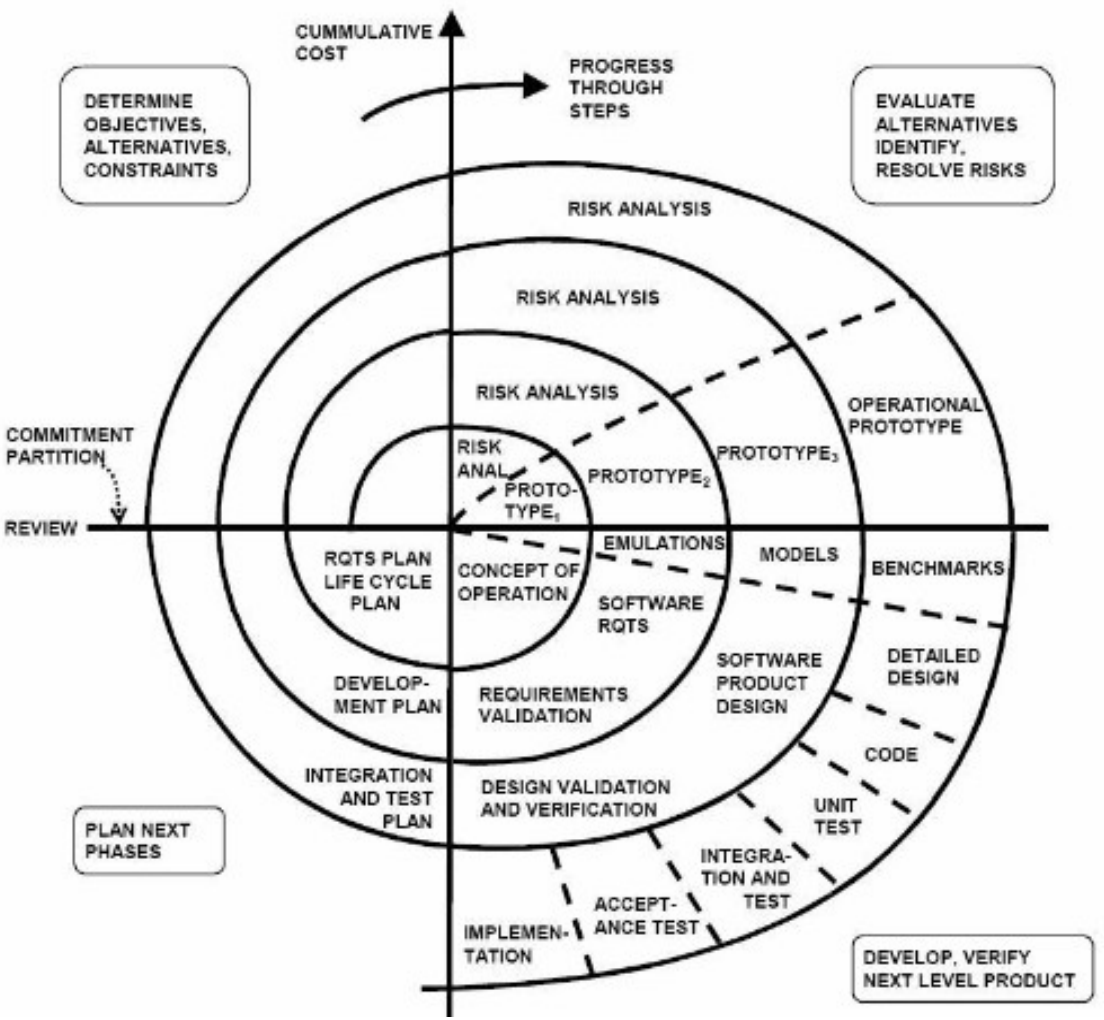
- Problematisch für große Projekte
- Softwareentwickler und Kunde verstehen sich nicht richtig
- Ergebnisse liegen erst zum Ende des Projekts vor
- Anforderungen können sich ändern

Prototypen-Modell



2 Arten von Prototypen:
„horizontal“ vs. „vertikal“

Spiralmodell



Spiralmodell Vorteile

- Man bekommt permanentes Feedback
- Man kann pro Zyklus, wenn man möchte ein anderes Prozess und Teammodell wählen
- Fehler werden relativ schnell erkannt
- Man hat bessere Eingriffsmöglichkeiten als bei einem Wasserfall

Spiralmodell Nachteile

- Man braucht für das Spiralmodell ein besseres Management
- Für kleine Projekte viel Aufwand

Das agile Manifest

- We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value
- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

Prinzipien

- Stelle den Kunden durch frühzeitige und regelmäßige Auslieferung nützlicher Software zufrieden
- Sich ändernde Anforderungen werden begrüßt, selbst wenn diese spät in der Entwicklung auftreten
- Lauffähige Software soll häufig, in Abständen von wenigen Wochen bis Monaten ausgeliefert werden
- Lauffähige Software ist das wichtigste Maß für den Projektfortschritt

Extreme Programming

- Grundsatzwerte
  - Kommunikation
  - Feedback
- Arbeitsmethoden
  - Kleine Releases
  - Testen
  - Refactoring
  - Paarweises Programmieren
  - Programmierrichtlinien

Praktiken

- Code Reviews sind gut
  - also werden Sie dauernd gemacht
  - und es gibt informelle Coding Standards
- permanentes Testen ist gut
  - also werden die Testfälle ständig auf dem Stand gehalten und aus geführt
  - JUnit, SUnit, Use Cases zugleich als Testscripts zu verwenden -> Unit Testing
- Integrationstests sind wichtig

Praktiken von XP sind nicht neu

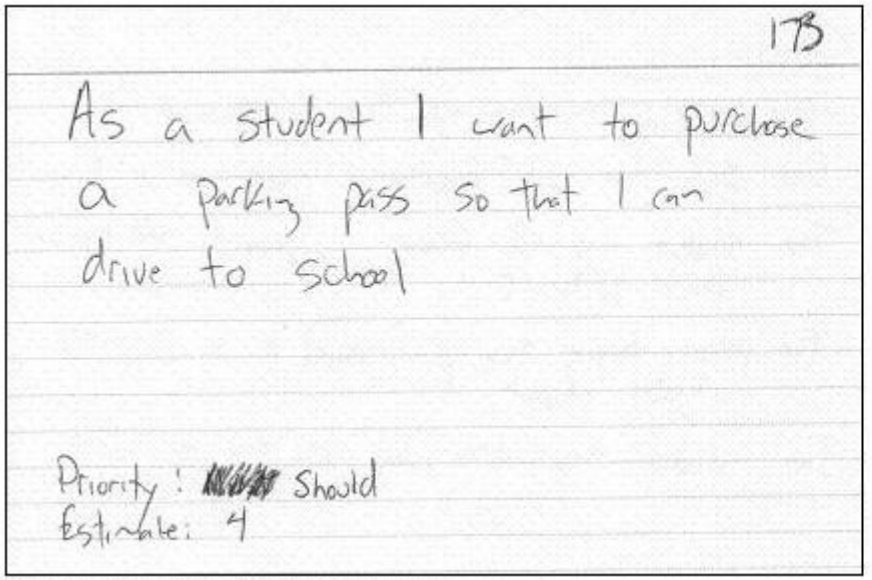
- Design ist wichtig
  - deshalb wird es während des Codierens durch Refactoring ständig verbessert
- Einfachheit ist gut
  - do the simplest thing that might possibly work
- Architektur ist wichtig
- Kurze Iterationen sind wichtig
- Feedback durch den Kunden ist wichtig
  - on site customer

Wie funktioniert

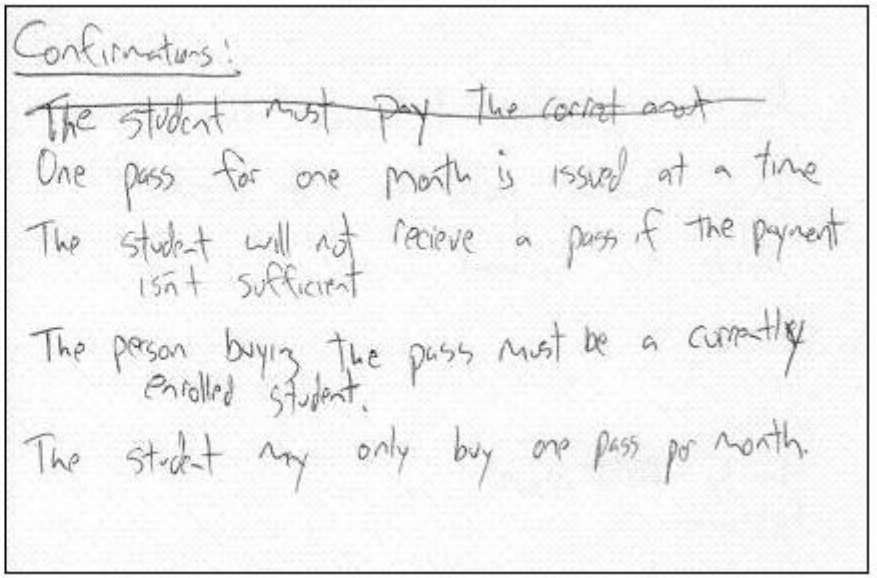
- Team sammelt Stories
- diese werden für das nächste Inkrement „priorisiert“
- und dann abgearbeitet
- nach einem Inkrement setzt man sich zusammen und macht das ganze von vorne

User Stories

Front of Card



Back of Card



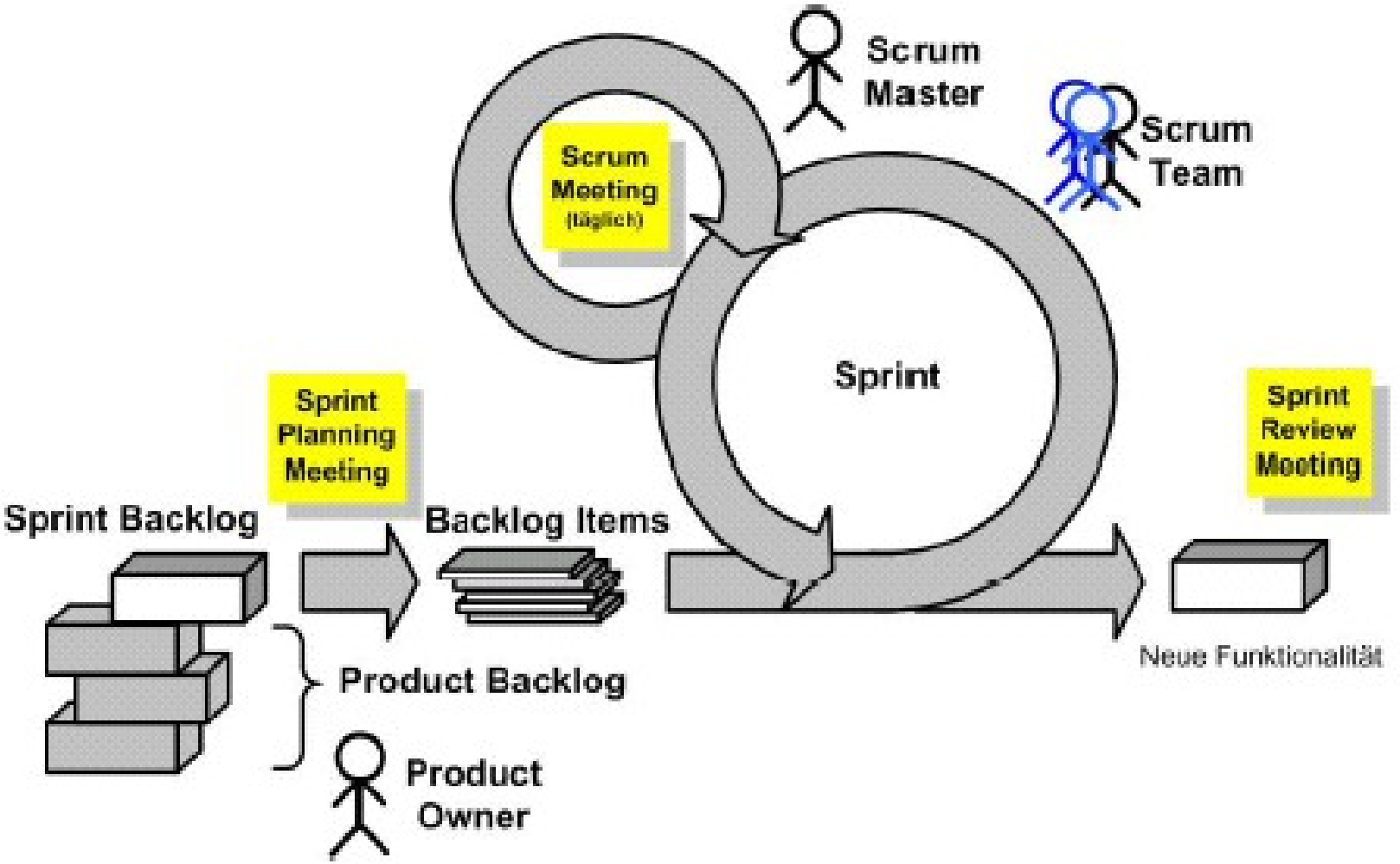
Crystal

- Familie von Vorgehensmodellen
  - Crystal Clear (bis 6 Mitarbeiter)
  - Crystal Yellow (bis 20 Mitarbeiter)
  - Crystal Orange (bis 40 Mitarbeiter)
- Inkrementelle Entwicklung - Releases alle 1-4 Monate
- Wenig Angaben zu detaillierten Prozessschritten
- Vertrauen in Selbstorganisation der Entwickler
- Regelmäßige Anpassungen des Prozesses durch die Mitarbeiter

Scrum

- Einfaches Modell, wenige Festlegungen
- Teams organisieren sich weitgehend selbst
- Entwicklungsprozess ist unterteilt in Sprints
- Vor jedem Sprint werden die zu entwickelnden Funktionalitäten festgelegt
- Tägliche Teambesprechung (Scrum)

Scrum

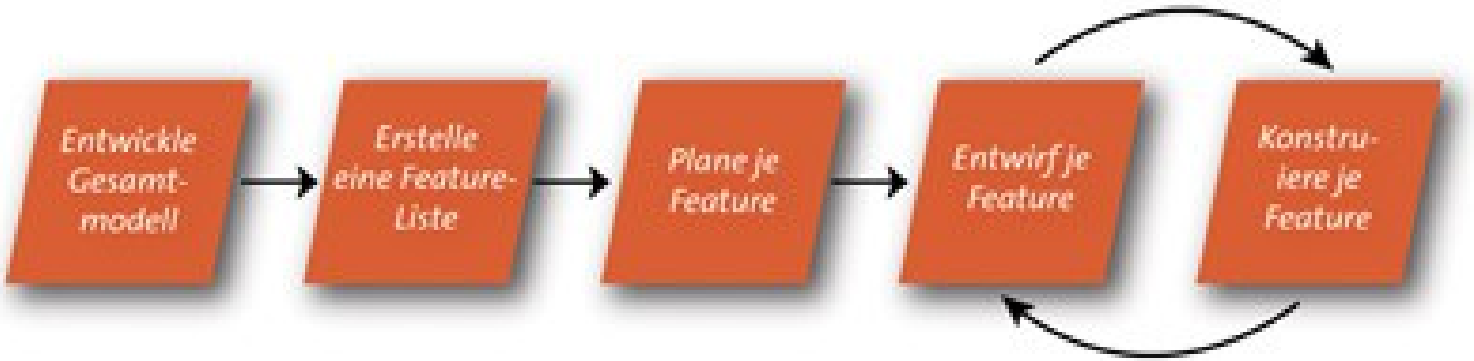

Scrum Master
Scrum Team
Scrum Meeting (täglich)
Sprint
Sprint Planning Meeting
Sprint Review Meeting
Sprint Backlog
Backlog Items
Product Backlog
Neue Funktionalität
Product Owner

Adaptive Software Development

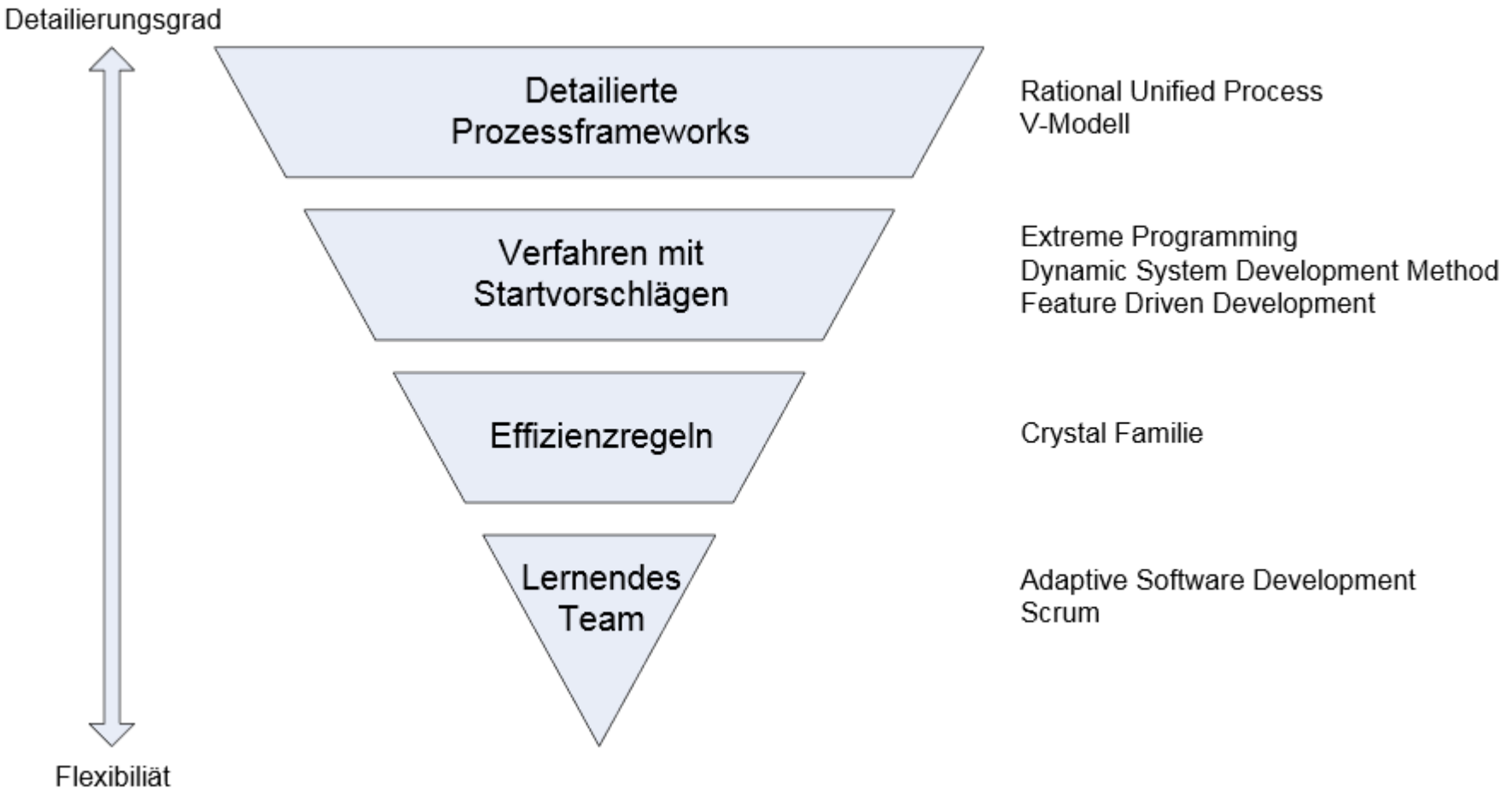
- Planen ist etwas paradoxes in einer adaptiven Umgebung
- 3 nicht-lineare überlappende Phasen
  - Spekulation
  - Zusammenarbeit
  - Lernen
- Resultate sind unvorhersagbar
- Änderungswünsche zeigen den Weg zum gewollten Ergebnis

Feature Driven Development

- Entwickeln eines Gesamtmodells
- Erstellen einer Feature-Liste
- Planung pro Feature
- Entwurf pro Feature
- Implementierung pro Feature



Einordnung agiler Methoden



Einschränkungen

Agile Methoden bieten nur eingeschränkte Unterstützung für

- verteilte Softwareentwicklung
- Entwicklung unter Beteiligung großer Teams
- Entwicklung sicherheitskritischer Software
- Konstruktion wiederverwendbarer Artefakte

SW-Entwicklung

- **Code & Fix**
  - Wenig/keine Planung
  - Viele kurzfristige Entscheidungen
  - nur für kleine Projekte geeignet
- **Software Engineering Methoden**
  - Entwicklung effizient und planbar machen
  - Festgelegte Vorgehensweisen (Prozesse)
  - Schwerpunkt auf Planung, Dokumentation
  - Bürokratisch
- **Agile**
  - Leichtgewichtig
  - Kompromiss zwischen zu viel Prozess und gar kein Prozess