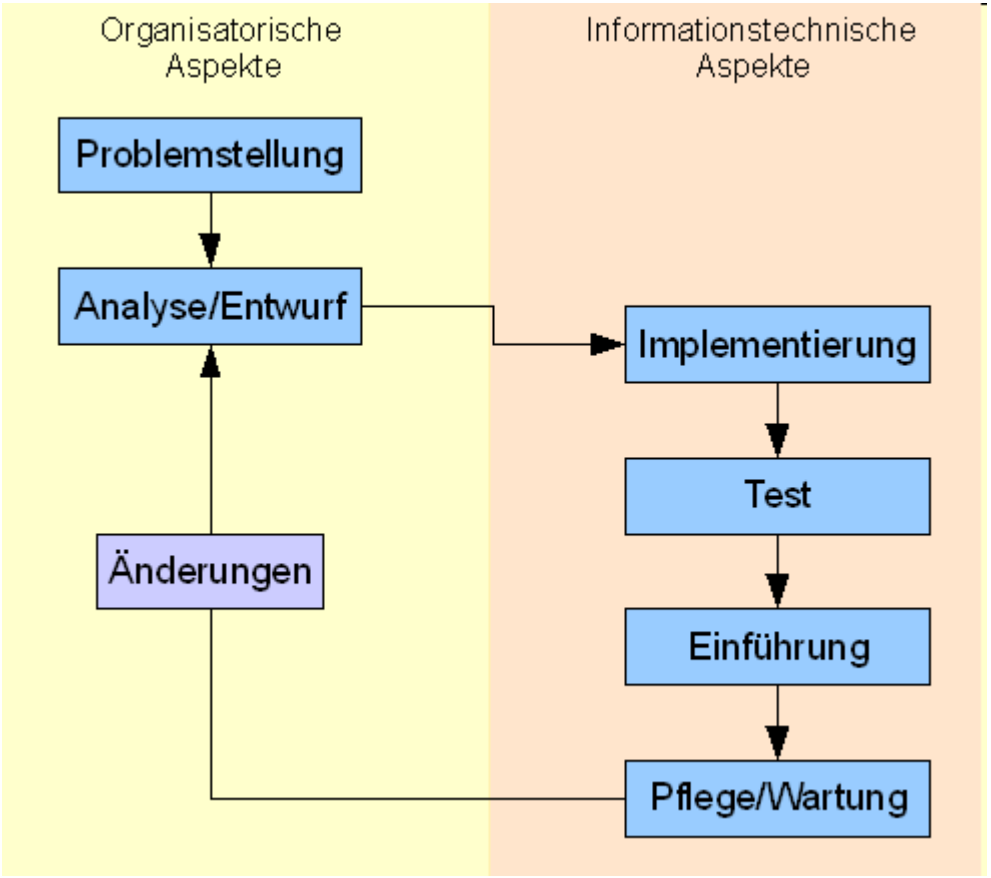
Software Engineering

Stephan Karrer

Lebenszyklus von Software



Quelle: http://wikipedia.de

Software entwickeln

- Wer erstellt die Problembeschreibung?
- Ist die Beschreibung ausreichend und korrekt?
- Welche Struktur und Eigenschaften soll das Programm haben?
- Ist das erstellte Programm eine Lösung des Problems?
- Was ist mit neuen Anforderungen an das Programm?
- Wie soll eine Aufgabenteilung für ein Entwicklerteam aussehen?
- …?

Was heißt eigentlich Software Engineering

- Software Engineering beschäftigt sich mit der ingenieurmäßigen Entwicklung von Software (wird in Deutschland teilweise auch Software-Technik genannt)

- Definition nach IEEE Computer Society:
  (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software
  (2) The study of approaches as in (1)

- Helmut Balzert: *Lehrbuch der Software-Technik:*
  „Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen.“

Warum Software Engineering

- Antwort auf die sogenannte Software-Krise:
  - Projekte laufen aus dem Zeitrahmen und/oder Kosten-Budget
  - Software ist ineffizient und von schlechter Qualität
  - Software entspricht nicht den Anforderungen
  - Projekte sind schwer zu managen
  - Code ist schwierig zu warten
  - Fehlende Produktivitäts- und Qualitätskontrolle

- Auch heute laufen Dinge aus dem Ruder:
  - Wahrscheinlich nur gut 50 % der IT-Projekte haben in Deutschland keine Abweichung bzgl. Budget, Zeit und Funktionalität *(Studie Success 2006)*
  - Weltweit scheitern möglicherweise 18% der IT-Projekte *(Chaos Report der Standish Group)*
  - Je größer das Projekt (Kosten, Zeit, Personen), desto größer die Herausforderungen

Komplexität der Software-Entwicklung

- Immenser Zuwachs an Rechen- und Speicherleistung
- Große, komplexe Software-Produkte (Millionen von Code-Zeilen)
- Integration in vorhandene System- und Software-Landschaft
- Viele Beteiligte und verschiedene, konkurrierende Anforderungen
- Anforderungen steigen und wechseln während des Lebenszyklus
- Vieles ist nicht ausreichend formalisierbar
- Große Projekte und Projektteams sind schwer zu managen
- Code ist nur bei entsprechender Modularisierung und Dokumentation wartbar
- Wissen und Erfahrung der Projekt-Mitarbeiter
- Korrekte Einschätzung von Kosten und Zeit
- ...

Versuch einer Definition: Komplexes Software-System

- Millionen Zeilen Source-Code
- Entwicklung mit vielen und wechselnden Personen
- Anhaltender, nachhaltender Einsatz

- ➢ Verständnis des Gesamtsystems ist für den einzelnen nicht mehr möglich
- ➢ Simultane Weiterentwicklung des Systems an verschiedenen Stellen
- ➢ Umbau als Dauerzustand

Ein typisches großes Entwicklungsprojekt

- 60-70 Anwendungskernobjekte, wie Kunde, Auftrag, Auftragsposition, Mitarbeiter etc.
- 150-200 Dialoge
- Es sind verschiedene Plattformen (Windows, UNIX, z/OS) und vorhandene Anwendungen einzubinden (Datenbanken, Schnittstellen zu anderen Software-Lösungen)
- Projektdauer ca. 2-3 Jahre
- Teamstärke 20-30 Mitarbeiter
- Potentiell hunderte Benutzer im Dialogbetrieb
- Entscheidend für das Kerngeschäft der Organisation, die das Projekt beauftragt hat

Erfolgsfaktoren

- Einbeziehung des Benutzers
- Unterstützung aus dem Management
- Klar spezifizierte Anforderungen und Ziele
- Genaue Planung
- Gute Organisation
- Hohe Kompetenz der Mitarbeiter
- Realistische Erwartungen/Schätzungen

➢ ***Der Methoden- und Sozialkompetenz kommt hohe Bedeutung zu !!***

Anforderungen an einen „Software-Ingenieur“

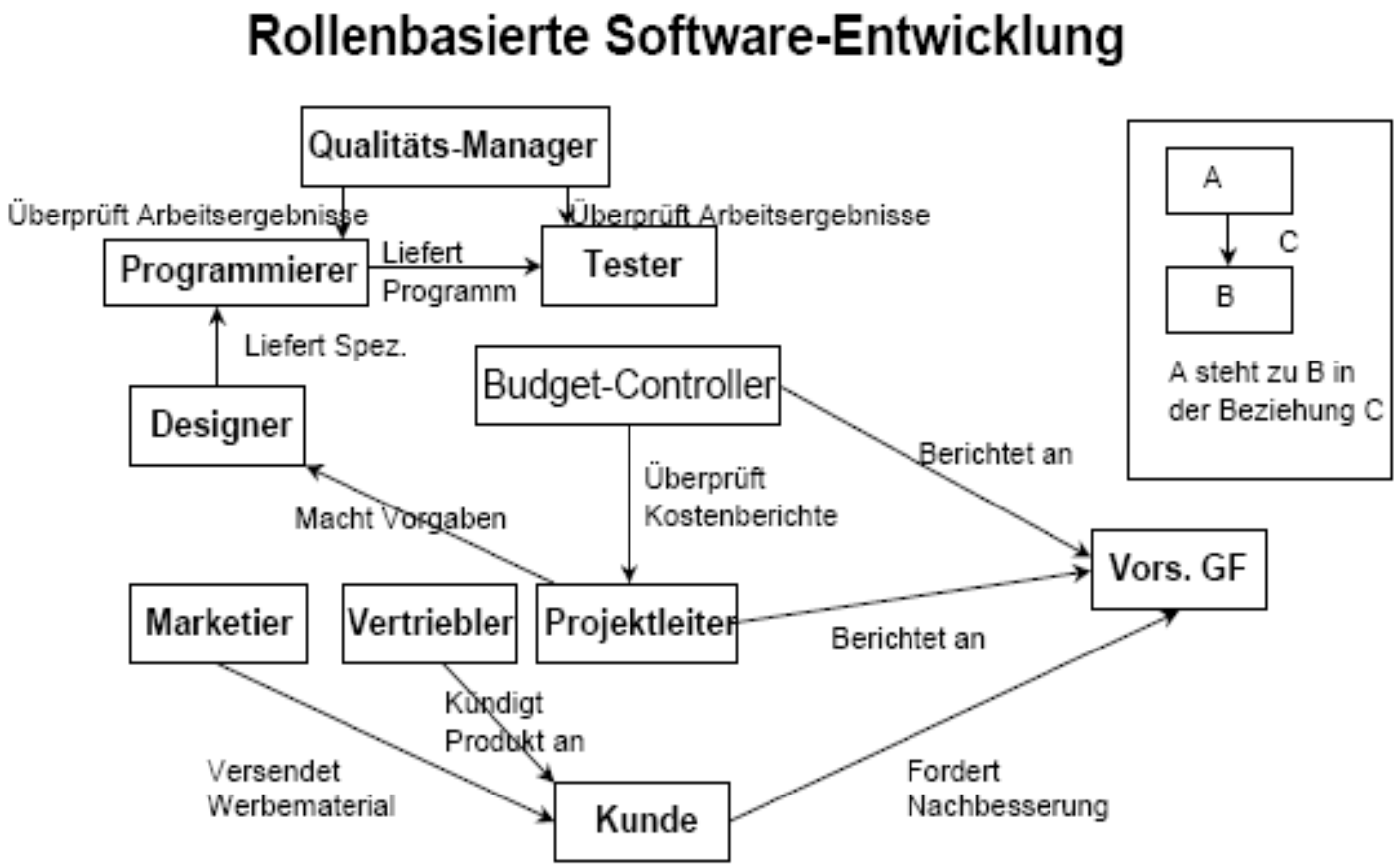
A software engineer must of course be a good programmer, be well-versed in data structures and algorithms, and be fluent in one or more programming languages. ... The software engineer must be familiar with several design approaches, be able to translate vague requirements and desires into precise specifications, and be able to converse with the user of a system in terms of the application rather than in ‚computerese‘.

Quelle: C. Ghezzi, M. Jazayeri, D. Mandrioli, Fundamentals of Software Engineering

Daraus folgende Fähigkeiten :

- Kommunikation auf verschiedenen Abstraktionsebenen
- Erstellung und Verwendung von Modellen / Spezifikationen
- Kommunikation mit Personen mit unterschiedlichen Zielsetzungen, Vorstellungen, Ausbildungen
- Arbeitsplanung und -koordination

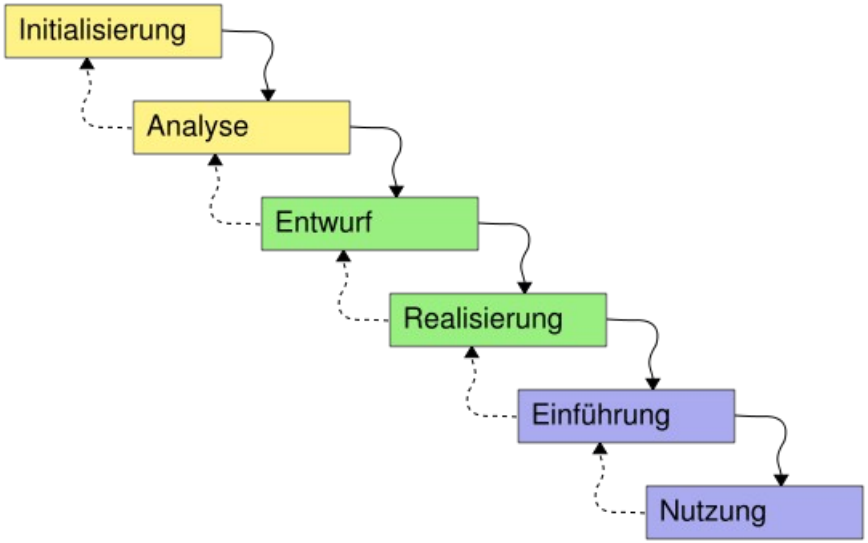
Rollen im Software-Entwicklungsprozess



Vorgehensmodelle

- Vorgehensmodelle oder auch Prozessmodelle unterteilen den Entwicklungsprozess in einzelne Phasen
- Der Entwicklungsprozess wird modularisiert und planbar
  - Leitfaden (Plan) für den Prozess existiert
  - In der Regel ist die projektbegleitende Dokumentation Bestandteil
  - Unabhängigkeit von einzelnen Personen
  - Einbettung von Test bzw. Abnahmephasen erlaubt frühzeitig Korrekturen
- Es gibt verschiedene Modelle, die alle ihre Vor- und Nachteile haben
- In der Praxis wird meist keine Reinform verwendet
- Der Projekterfolg, nicht das Vorgehensmodell, ist das Ziel

Das Wasserfallmodell



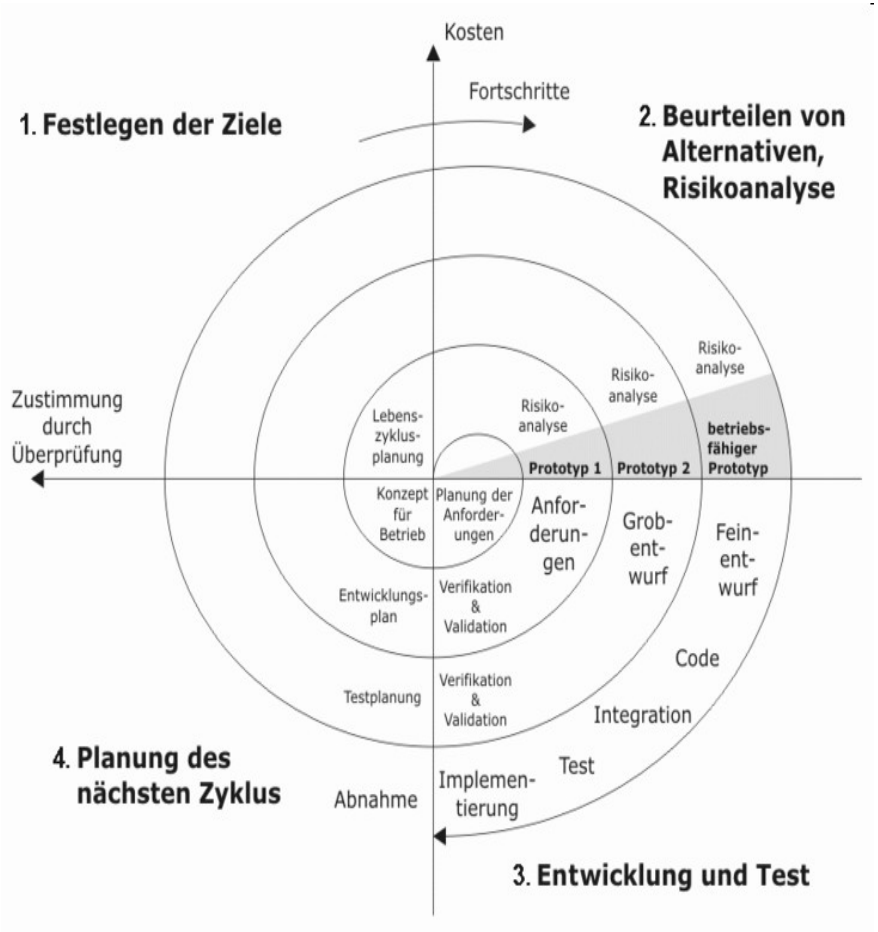
Quelle: http://wikipedia.de
Autor Maciej Jaros, veröffentlicht unter GNU-Lizenz für freie Dokumentation (GFDL)

- Ursprung 1970 von W. Royce, spätere Varianten von B. Böhm
- Die Phasen werden streng sequentiell durchlaufen
- Jede Phase hat ein klar definiertes Ergebnis (dokumentiert)
- Die nächste Phase wird erst begonnen, nachdem die vorhergehende validiert und abgeschlossen ist
- Spätere Modifikationen erlauben auch den Rücksprung in eine frühere Phase

Merkmale des Wasserfall-Modells

- Das Wasserfallmodell ist an den Bedürfnissen des Managements ausgerichtet:
  - Klare Phasenenden mit eindeutig zugeordneten Ergebnissen
  - Klare Verantwortungen für Phasen
  - Einsatz traditioneller Management-Methoden (Meilensteine, Qualitätssicherung per Zwischenergebnis)
  - Dokumentengetrieben
- Die Trennung der Phasen und der sequentielle Ablauf gehen an der Entwicklungspraxis vorbei
  - Auftraggeber haben keine genaue Vorstellung des Systems
  - Erkenntnisgetriebene Entwicklung
  - Phasen überlappen in der Regel
  - Verantwortung wird mit jeder Phase abgegeben an die folgende Phase
  - Effekt „Lieferung des Bestellten“
  - Testen nur in einer Phase am Ende ist kritisch
  - Nichteinhaltung der Zeitplanung verschiebt alles, in der Regel zu Lasten der Testphase

Das Spiralmodell



Quelle: http://wikipedia.de
Autor Saibo, veröffentlicht unter GNU-Lizenz für freie Dokumentation (GFDL)

- Ursprung 1986 von B.W. Boehm
- Das Modell geht davon aus, dass wir uns der Lösung iterativ nähern
- In jedem Iterationsschritt werden die relevanten Phasen neu durchlaufen, um zu einem besseren Ergebnis zu kommen
- Folgende 4 Schritte bilden im ursprünglichen Modell einen Zyklus:
  - Ziele, Randbedingungen identifizieren
  - Evaluierung der Alternativen, Risiken
  - Entwicklung und Test von Teilergebnissen
  - Erweiterung bzw. Detaillierung der Spezifikation/Planung

Merkmale des Spiralmodells

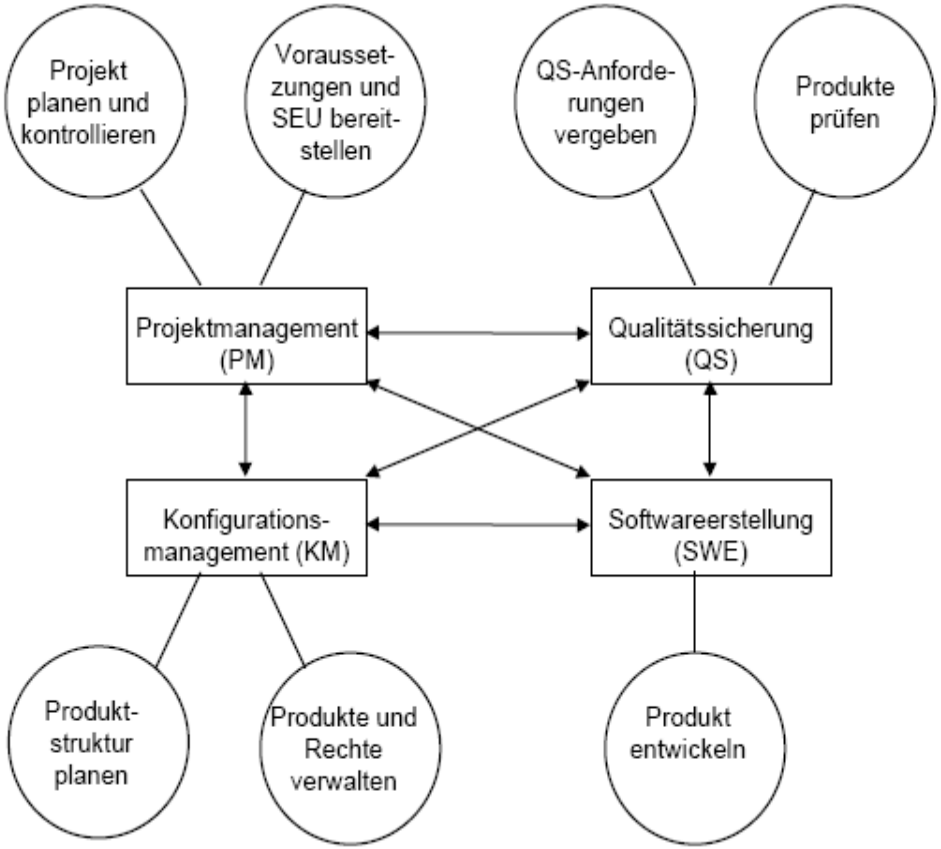
- Vorteile
  - Entstehen von Prototypen
  - Bessere schrittweise Abstimmung zwischen den Beteiligten
  - Überprüfung in regelmäßigen Intervallen, u.U. erneute Festlegung des Prozeßablaufs in Abhängigkeit von den Risiken
  - Ein Prozeß-Modell wird nicht für die gesamte Entwicklung festgelegt
  - Fehler und ungeeignete Alternativen werden frühzeitig eliminiert

- Nachteile
  - schwierige Risiko- und Kostenabschätzung
  - schwierige Zeitabschätzung
  - Hoher Managementaufwand wegen häufiger Entscheidungen über weiteren Prozeßablauf
  - weniger für kleinere und mittlere Projekte geeignet

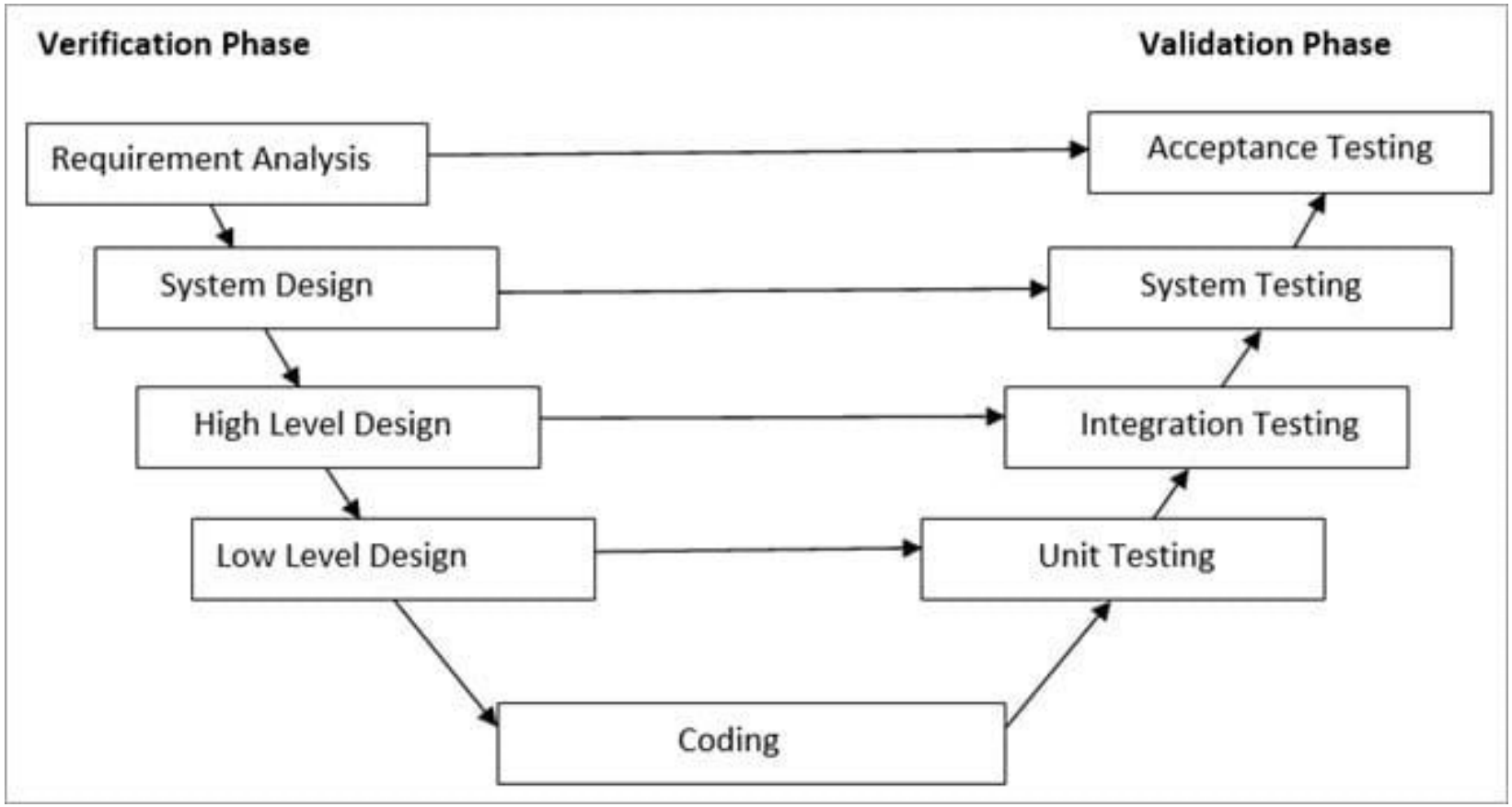
Das V-Modell

- Lebenszyklusmodell, das ursprünglich für die Software-Entwicklung im Bereich der Bundeswehr entwickelt wurde
- Ist heute üblicher Standard bei Verwaltung von Bund und Ländern und wird auch bei großen Unternehmen eingesetzt
- Aktuelle Variante ist V-Modell XT (Extreme Tailoring): Flexiblere Anpassung an Projektgrößen durch Minimierung einzelner Phasen
- Wird mittlerweile für eine Vielzahl von öffentlichen Entwicklungsprojekten gefordert
- Der Name resultiert aus der schematischen Anordnung der wesentlichen Entwicklungsaktivitäten im Bereich der Software-Erstellung.
- Die grundlegende Idee ist die enge Integration von
  - Software-Erstellung
  - Qualitätsmanagement
  - Projektmanagement
  - Konfigurationsmanagement

Tätigkeitsbereiche im V-Modell



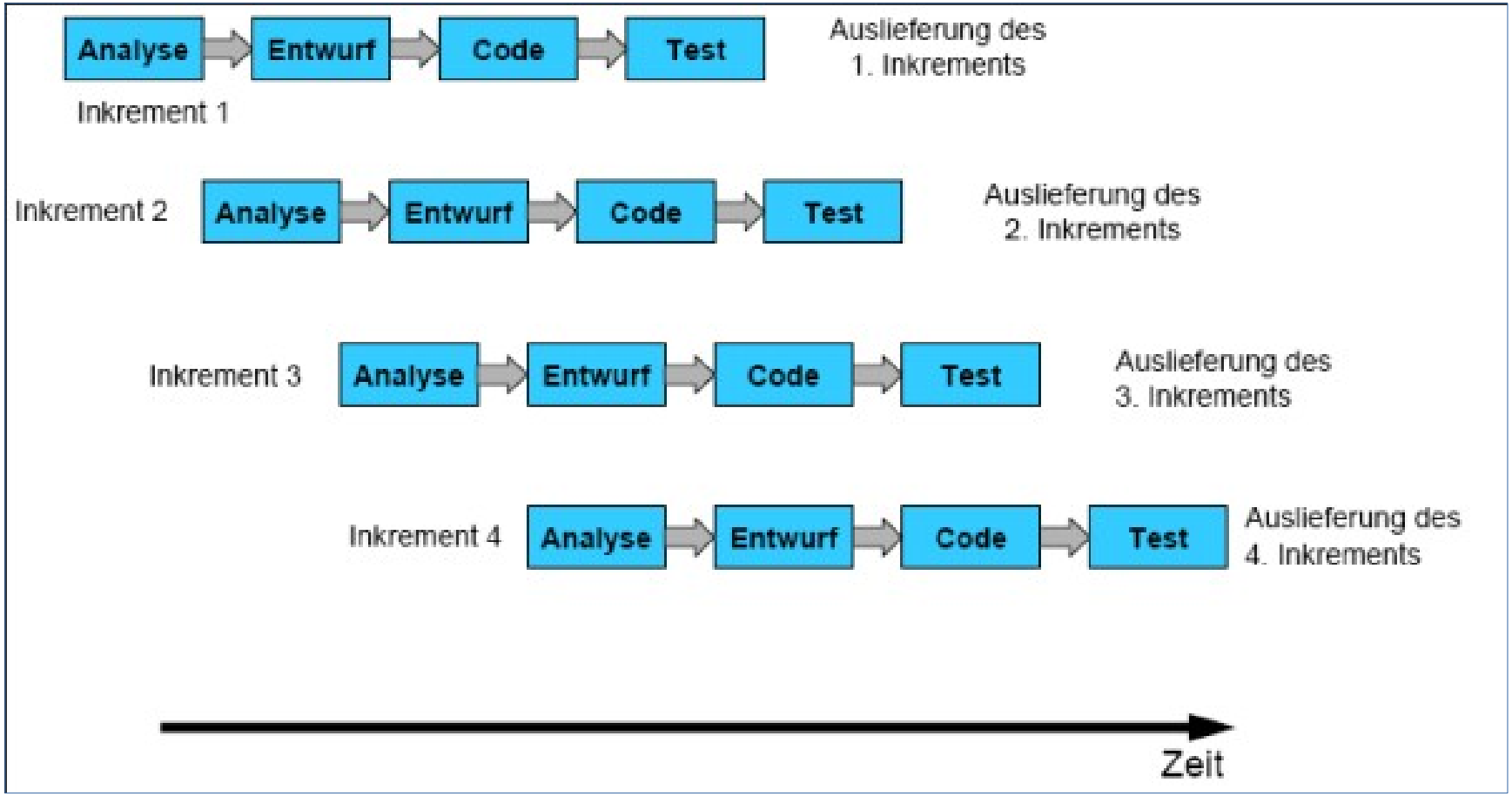
Das V-Modell:
Testen findet auf jeder Ebene statt



Merkmale des V-Modells

- Vorteile
  - Berücksichtigt wesentliche Elemente industrieller Software-Entwicklung (Qualitätssicherung, Systemerstellung, Konfigurationsmanagement, Projektmanagement)
  - Identifikation von Rollen und Verantwortungen
  - Durchgängige Tests in der Software-Entwicklung
  - Flexibilität: Anpassbar an verschiedene IT-Problemdomänen
  - Gut geeignet für Großprojekte
- Nachteile
  - Sehr hoher Managementaufwand
  - Skalierbarkeit: ursprünglich für kleine / mittlere Projekte ungeeignet, dank XT verbessert
  - Sehr komplex

Inkrementelles bzw. iteratives Vorgehensmodell



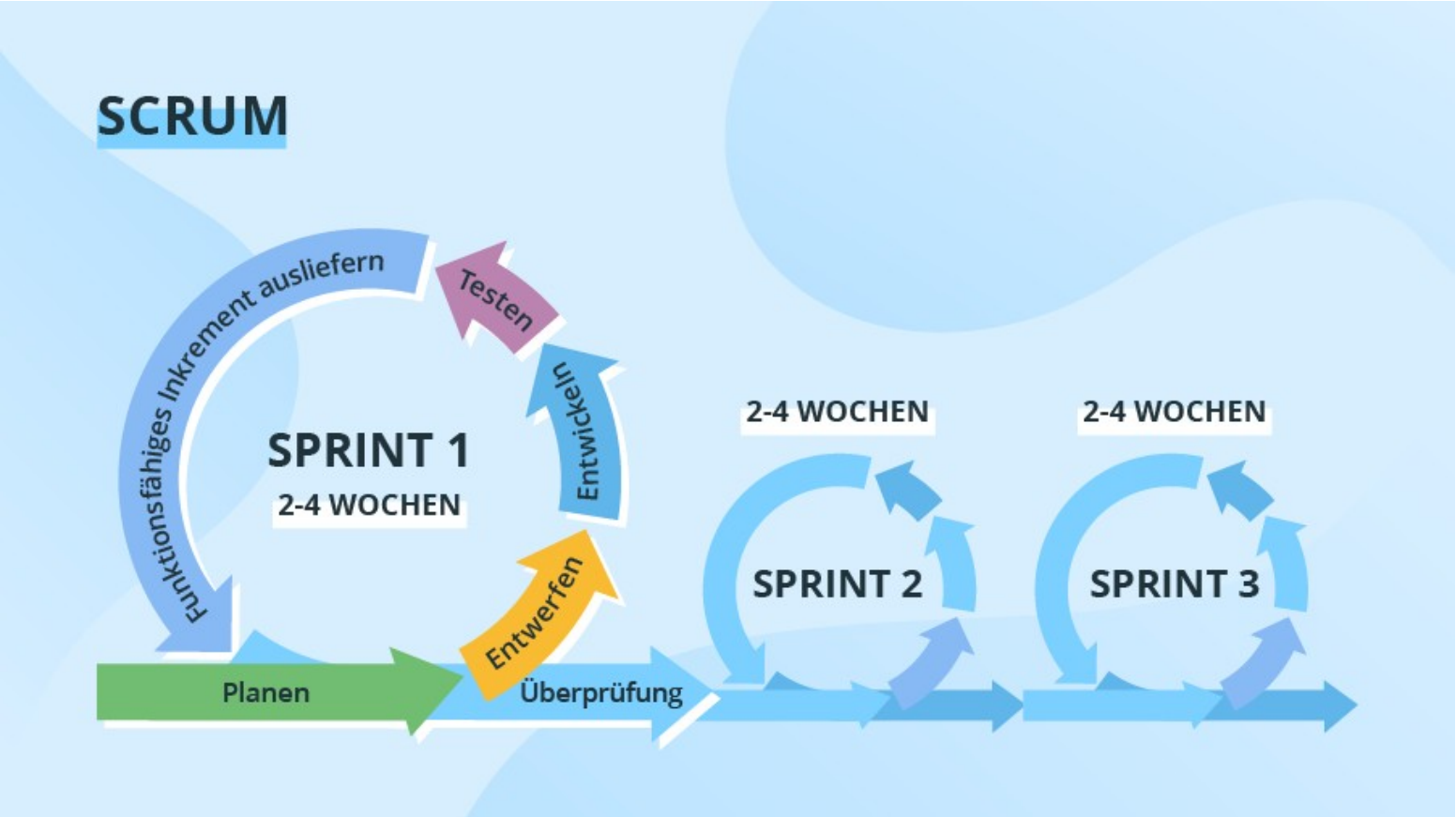
Merkmale des inkrementellen Modells

- Die Entwicklung wird in Schritte unterteilt, die jeweils einen Teil der geforderten Funktionalität realisieren. Nachfolgende Schritte können sowohl neue Funktionalität realisieren als auch vorhandene erweitern (Iterativ).

- Vorteile
  - Priore Anforderungen werden in ersten Schritten realisiert
  - Anforderungen können kontinuierlich weiterentwickelt werden, keine vollständige Spezifikation zu Beginn erforderlich
  - Schritte können potentiell parallel ausgeführt werden
  - Feedback des Kunden kann früh berücksichtigt werden

- Nachteile
  - Anforderungen müssen sich in Schritte zerlegen lassen (Modularisierung)
  - Gesamtaufwand und Kosten lassen sich vorab schwierig einschätzen

Agile Vorgehensmodelle (Scrum, Extreme Programming, ...)



Merkmale der agilen Modelle

- Weiterentwicklung der inkrementellen Vorgehensweise
  - Kurze Schritte (Sprints)
  - Frühzeitiges Feedback
  - Intensive Kommunikation
  - Wird aktuell in der Regel stark genutzt
  - Erfordert entsprechende Kommunikationsstrukturen
  - Erfordert Tool-Unterstützung, manuelle Umsetzungen sind häufig zu langsam

DevOps ??

- Der Begriff setzt sich aus „Dev“ (Development, Entwicklung) und „Ops“ (Operations, Vorgänge) zusammen
- Ist eher Philosophie als Vorgehensweise
- Fokus auf optimaler Verzahnung von Software-Entwicklung und IT-Betrieb
- Häufiges Ziel: Schnellere und stabilere Software-Bereitstellung ohne den Betrieb zu überfordern

Weiterführende Informationen

- Studien:
  - Chaos Chronicles, Standish Group International:
    Seit 1994 jährlich herausgegebener Bericht zum Erfolg von IT-Projekten in den USA http://www.standishgroup.com
  - Success Studie 2006, OFFIS e.v.:
    Eine Umfrage im Auftrag des BMBF zu Erfolgsfaktoren von IT-Projekten in Deutschland
    https://www.offis.de/offis/publikation/success-erfolgs-und-misserfolgsfaktoren-bei-der-durchfuehrung-von-hard-und-software-entwicklungsprojekten-in-deutschland.html
- Literatur:
  - Helmut Balzert: Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb. Springer, 3.Auflage, 2011
  - C. Ghezzi, M. Jazayeri, D. Mandrioli, Fundamentals of Software Engineering, Prentice-Hall, 1991
- Informationen im Web:
  - Portal der IEEE Computer Society zum Thema Software Engineering
    http://www.computer.org/portal/site/seportal