Softwaretechnik und Programmierparadigmen

VL02: Softwareentwicklungsmodelle

Prof. Dr. Sabine Glesner
FG Programmierung eingebetteter Systeme
Technische Universität Berlin

Wozu Entwicklungsmodelle?

- Softwareentwicklung ist mehr als nur programmieren!
  - Standardisierte Prozesse sind notwendig
  - Strukturierung des Vorgehens notwendig
  - Verlässliche Planung notwendig
  - Vergleich und Bewertung von Projekten
- Systematisches Vorgehen durch
  - Planen
  - Entwerfen
  - Entwickeln
  - Testen
  - Warten

Wozu Entwicklungsmodelle?

- Entwicklungsmodelle:

  Koordination von Arbeitsschritten
  - Mit welchen Aktivitäten
  - In welcher Reihenfolge
  - Von welchen Personen
  - Mit welchen Ergebnissen
  - Mit welchen Qualitätssicherungsmaßnahmen

Wesentliche Arbeitsschritte in der Softwareentwicklung

- Anforderungsermittlung
- Analyse der Anforderungen
- Entwurf
- Implementierung
- Test/ Qualitätssicherung
- Inbetriebnahme, Wartung, Evolution

Keine fixe Reihenfolge dieser Aktivitäten!

→ Softwareentwicklungsmodell

Gibt es ein bestes Entwicklungsmodell?

- Nein, denn abhängig von
  - Größe des Entwicklungsteams
  - Größe des zu entwickelnden Produkts
  - Konstanz der Anforderungen
  - Einsatzbereich
    - sicherheitskritisch?
  - Rechtliche Fragen
    - Einhaltung von Standards, Umfang der Dokumentation
  - Lebensdauer der Software

Plan-basierte vs. Agile Methoden

- Plan-basierte Methoden
  - Alle Aktivitäten im Vorfeld geplant
  - Trennung der Arbeitsschritte und explizite Dokumente zur Kommunikation zwischen diesen
  - z.B. Wasserfallmodell, V-Modell, (Rational) Unified Process
- Agile Methoden
  - Entwurf und Implementierung liegen im Fokus, andere Arbeitsschritte werden zum Teil mit eingewoben
  - Wesentliche Motivation: Reagieren auf Anforderungsänderungen
  - z.B. Extreme Programming, Feature Driven Development, Scrum

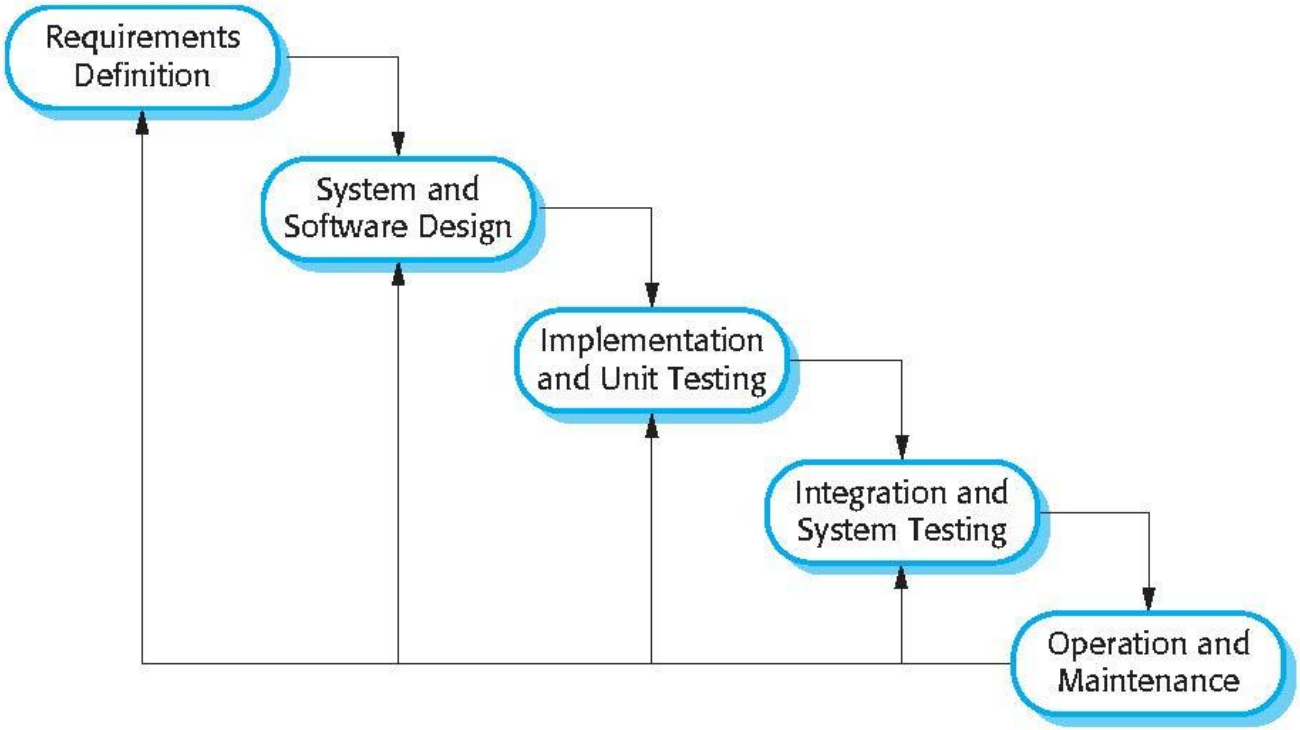
Überblick über Vorlesung

- Plan-Basierte Entwicklungsmodelle
  - Wasserfall-Modell
  - V-Modell
  - Spiralmodell
  - (Rational) Unified Process
- Agile Entwicklungsmodelle
  - Extreme Programming
  - Scrum

Überblick über Vorlesung

- Plan-Basierte Entwicklungsmodelle
  - Wasserfall-Modell
  - V-Modell
  - Spiralmodell
  - (Rational) Unified Process
- Agile Entwicklungsmodelle
  - Extreme Programming
  - Scrum

Wasserfallmodell

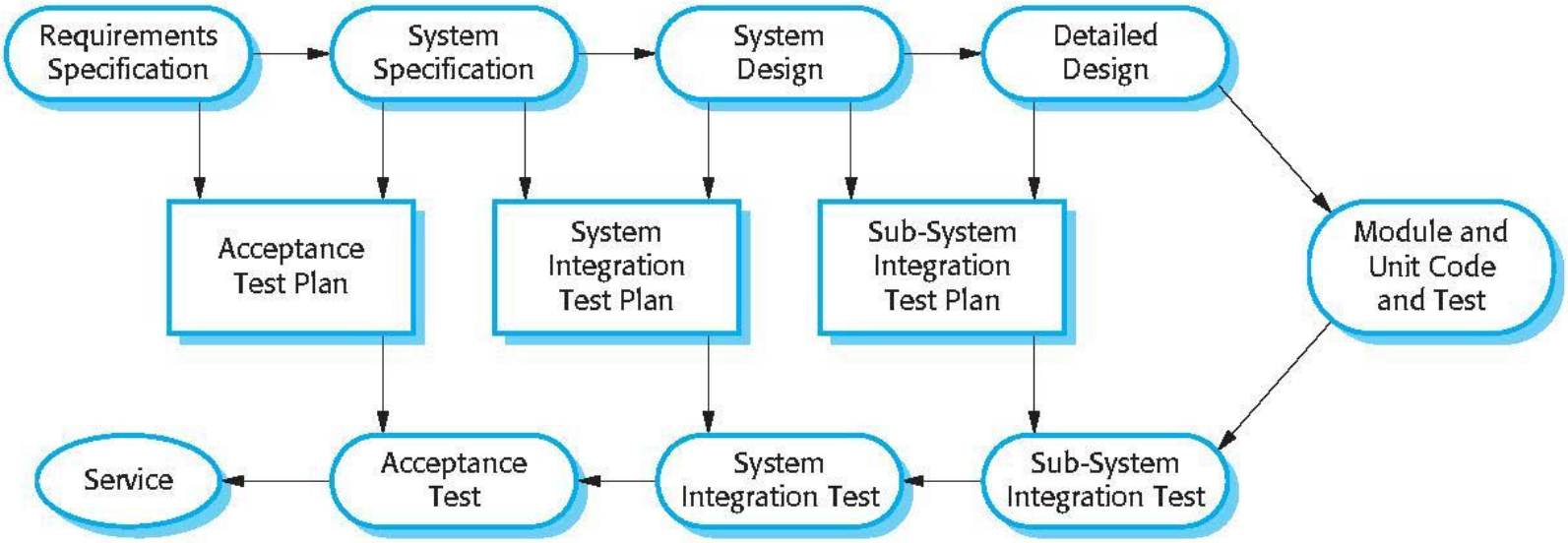


Wasserfallmodell

- Sequentielle Ausführung der Arbeitsschritte
- ggf. Überarbeitung vorheriger Schritte

- Vorteile
  - Klare Abgrenzung der Arbeitsschritte
  - Einfache Planung und Steuerung
- Nachteile
  - Planung aufwändig (muss u.U. mehrfach erfolgen)
  - Stabile Anforderungen benötigt
  - Einführung des Systems spät nach Entwicklung
  - Späte Entdeckung von Fehlern (Trennung von Implementierung und Test)

V-Modell



Idee: frühere Erkennung von Fehlern möglich

V-Modell

- Weiterentwicklung des Wasserfallmodells
- Relevanz von Qualitätssicherung/Testen wird betont

- Vor- und Nachteile
  - Ähnlich wie bei Wasserfallmodell, außer dass eine höhere Qualität zu erwarten ist

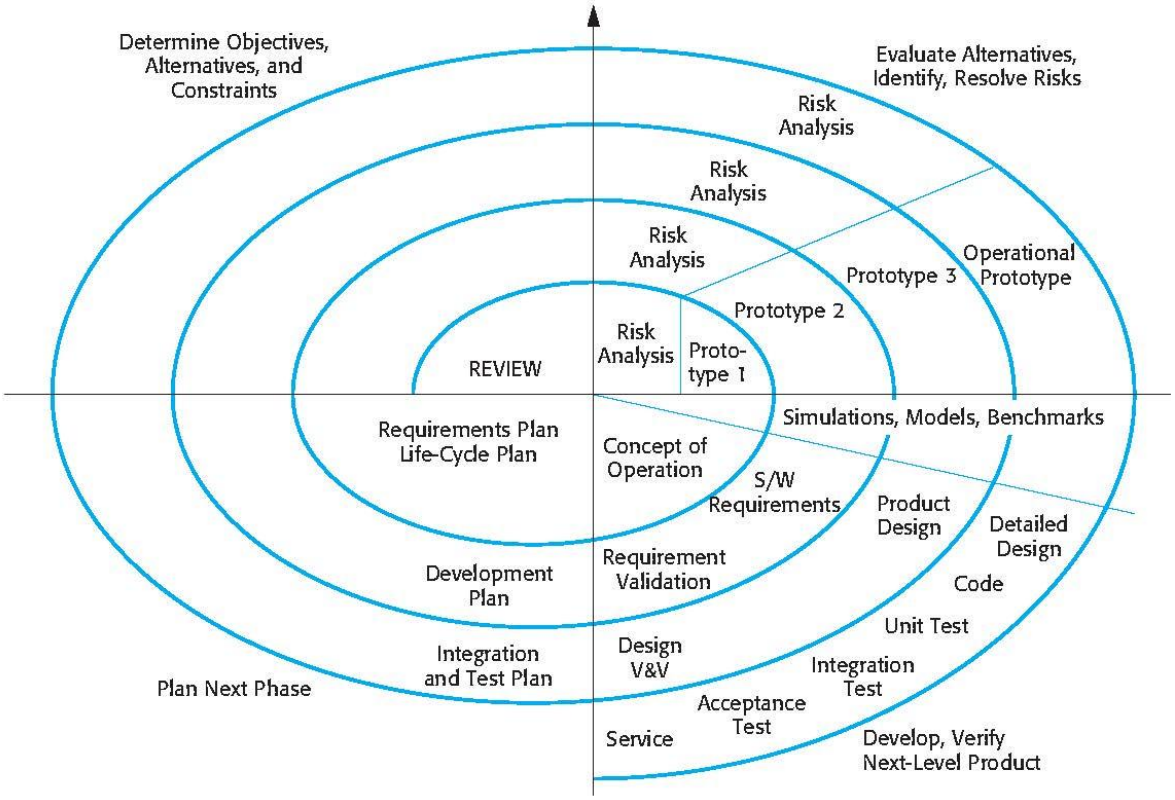
Inkrementelle Entwicklung allgemein

- Basisaktivitäten des Software-Engineering (Analyse, Entwurf, Entwicklung, Testen) treten verschachtelt und wiederholt auf
- Verschiedene Inkremente können bereits getestet werden und an Kunden gegeben werden
- Frühes Feedback durch Tests und Feedback durch Kunden
  - Frühzeitige Validierung und Verifikation

Grundsätzlich: Keine Verfeinerung eines laufenden Systems eingeplant: wenn System läuft (Plan abgearbeitet), ist es fertig
-> In der Praxis aber häufig nötig (neue Anforderungen, erste Erfahrungen)

Spiralmodell

Beginn in der Mitte, Wiederholung der Schritte:
Analyse
Konzept
Prototyp



Text

Spiralmodell

- Iteratives Entwicklungsmodell mit den Aktivitäten:
  - Zielbeschreibung
  - Risikoanalyse und Prototyping
  - Erstellung und Evaluierung eines Zwischenprodukts
  - Planung des nächsten Zyklus

Planung steht immer noch vor Durchführung

- Vorteile
  - Iterative Entwicklung reduziert Risiko
  - Frühes Testen (auch durch Benutzer) durch Prototyping
- Nachteile
  - Eher für große Projekte geeignet
  - Risikoanalyse kann teuer sein

(Rational) Unified Process

- Erstmals 1999 veröffentlicht von Jacobson, Booch und Rumbaugh (Rational Software – 2003 aufgekauft von IBM)
- Erweiterbarer abstrakter Entwicklungsprozess für viele verschiedene Situationen einsetzbar
- Orthogonale zeitliche Aufteilung in Entwicklungsphasen
  - Inception Was möchte ich?
  - Elaboration Ausarbeitung
  - Construction Bau / Umsetzung
  - Transition
- In jeder dieser Phasen werden (iterativ) die Arbeitsschritte bearbeitet

Rational Unified Process

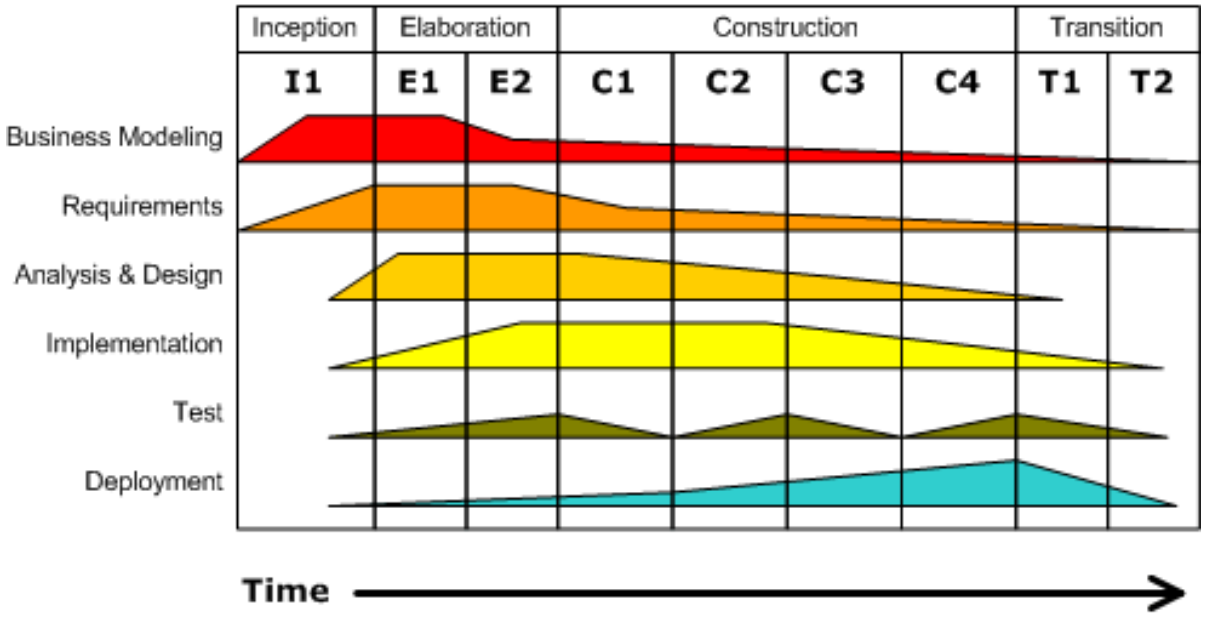
- **Kernarbeitsschritte**
  - Geschäftsprozess-modellierung
  - Anforderungsanalyse
  - Analyse und Design
  - Implementierung
  - Test
  - Auslieferung

Schritte, die in einer Iteration durchgeführt werden

- **Unterstützende Arbeitsschritte**
  - Konfigurations- und Änderungsmanagement
  - Projektmanagement
  - Infrastruktur

Änderungsmanagement:
Änderungen nachvollziehbar machen (wichtig für große Projekte, großen Teams) bzw. gut dokumentieren -> Tests können eingespart werden

Rational Unified Process



RUP Best Practises

RUP als Zusammenfassung von Best Practices (basierend auf Erfahrung)

- Iterative Entwicklung — Idee verfolgen und immer weiter anpassen
- Anforderungsmanagement
- Verwendung von Komponenten (Reuse)
- Visuelle Modellierung
- Qualitätssicherung
- Änderungsmanagement

Exkurs: IEC 61508

- Internationale Norm zur Entwicklung von sicherheitskritischen Systemen
- Generisch gehalten
- Wurde 1998 erstmals veröffentlicht
- Definiert Safety Integrity Levels (SILs)
  - SIL1 – verhältnismäßig geringer Aufwand für Risikovermeidung - bis SIL4 – hoher Aufwand für Risikovermeidung

Unterscheidung zwischen sicherheitskritischen und weniger wichtigen/gefährlichen Komponenten (Bremse vs. Radio im Auto)

Exkurs: ISO 26262 („Road vehicles – Functional Safety“)

- ISO-Norm zur Gewährleistung der funktionalen Sicherheit von elektronischen Komponenten im Automobil
- Anpassung von IEC 61508 für Automobilbereich
- Seit 2011 in Kraft
- Freiwillig, aber immer mehr Hersteller bestehen auf Einhaltung von Zulieferern
- Definiert ein Vorgehensmodell für die Entwicklung von Automobil-Software

Überblick über ISO 26262

1. Vokabular
2. Management der funktionalen Sicherheit
3. Konzeptphase
4. Produktentwicklung: Systemebene
5. Produktentwicklung: Hardwareebene
6. Produktentwicklung: Softwareebene
7. Produktion, Betrieb und Außerbetriebnahme
8. Unterstützende Prozesse
9. ASIL- und sicherheitsorientierte Analysen
10. Guideline (nur informativ)

Wichtige Bestandteile der ISO 26262

- Produktentwicklungsteile
  - Detailiertes Plan-basiertes Vorgehen (V-Modell basiert)
- Automotive Safety Integrity Level (ASIL)
  - Abstrakte Klassifikation von Levels von Sicherheitsrisiken
    - Level der benötigten Risiko-Reduktion zum Vermeiden von Gefahren (Stufen A bis D)
  - Resultat der Hazard Analysis und Risk Assessment Wahrscheinlichkeit für Eintritt eines Ereignisses, Häufigkeit, Schwere
  - Teilkriterien
    - Severity Classification (S)
    - Exposure Classification (E)
    - Controllability Classification (C) Steuerbarkeit
- Safety Life Cycle
  - Vollständige Menge der Gefahren-Events bzgl Komponente identifizieren
  - ASIL für jedes solche Event
  - Sicherheitsziel für jedes Event definieren
  - Architektur zur Einhaltung der Sicherheitsziele entwerfen
  - Sicherheitsziele in Sicherheitsanforderungen verfeinern
  - Implementierung und Verifikation bzgl Sicheheitsziele und Sicherheitsanforderungen

Überblick über Vorlesung

- Plan-Basierte Entwicklungsmodelle
  - Wasserfall-Modell
  - V-Modell
  - Spiralmodell
  - (Rational) Unified Process
- Agile Entwicklungsmodelle
  - Extreme Programming
  - Scrum

Agiles Vorgehen

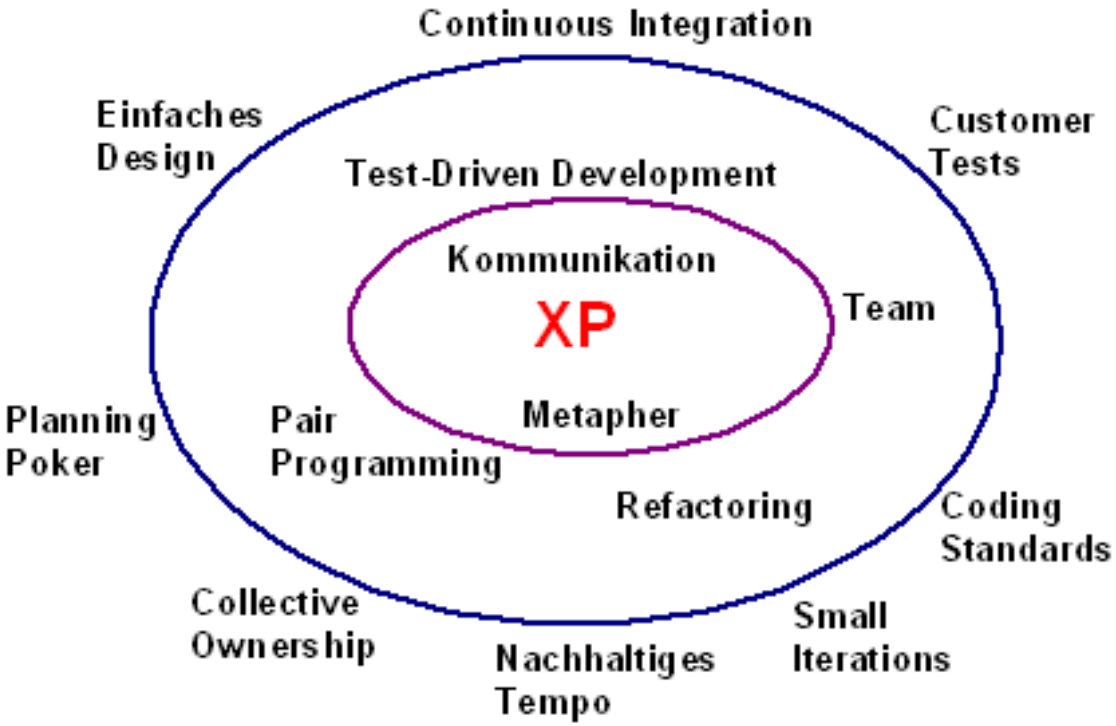
- Inkrementelle Softwareentwicklung
- Schnelle Entwicklung von (inkrementellen) Prototypen
- Einbeziehung des Auftraggebers in die Entwicklung
- Häufige Änderungen in den Requirements erwartet
  - Daher: Entwicklung nur der nötigsten Modelle/ Dokumente
- Unterschiedliche Fähigkeiten der Entwickler werden besser berücksichtigt
  - „People not Process!“

nicht klassische Ingenieur-Sicht (sorgfältig Planen vor Durchführung)

Extreme Programming

[http://s3.amazonaws.com/giles/hawaii_051109/extreme.jpg]

Extreme Programming – Best Practices



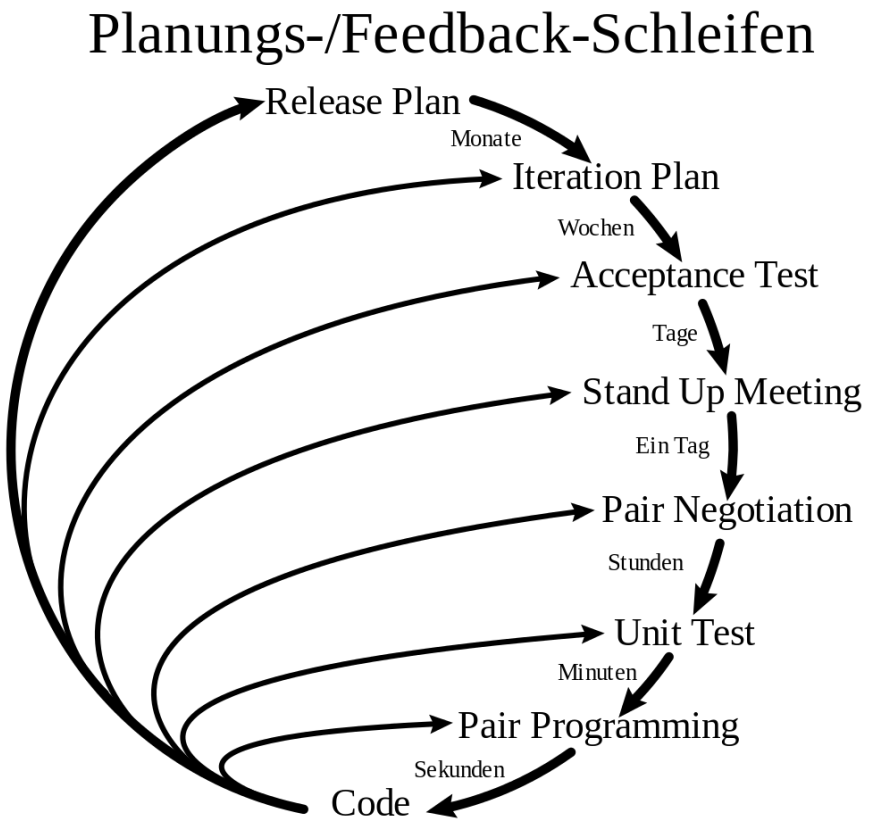
[http://upload.wikimedia.org/wikipedia/commons/thumb/7/7e/Xp-kreis.png/400px-Xp-kreis.png]

Extreme Programming – Best Practices (Auszug)

nicht planbasiert, dafür mehr Verantwortung für einzelne Teammitglieder

- Iterative Entwicklung
  - Vor allem kurze Iterationen, um Kunden lauffähige Zwischenzustände zeitnah präsentieren zu können
- Pair Programming zu zweit an einem Rechner programmieren
  - Ständiges Code-Review
  - Ständiges Refactoring
  - Gemeinsame Verantwortung
- Kollektives Eigentum
  - Einzelne Teammitglieder besitzen kein alleiniges Wissen über Komponente (z.B. durch Pair Programming)
  - Wechselnde Einsatzgebiete der Teammitglieder
- Test-Driven Development
  - Erst Tests dann Programm

Extreme Programming – Feedback Loops



[http://upload.wikimedia.org/wikipedia/commons/2/28/Extreme_Programming_Planungs-_und_Feedback-Schleifen.svg]

Scrum

"Gedränge"

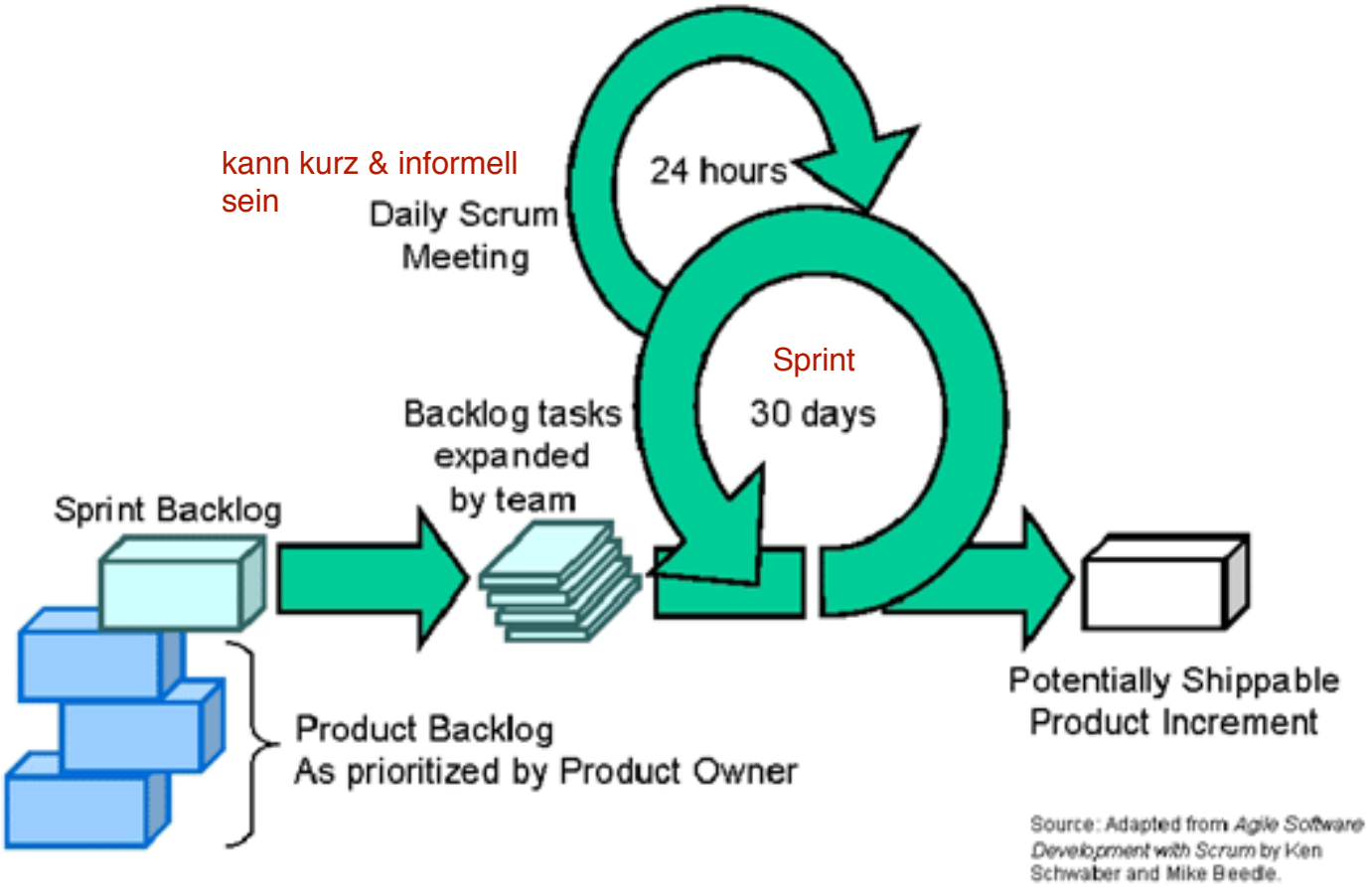
- Framework für das Projekt-Management
- Vorgehensmodell, das sehr gut mit Extreme Programming harmoniert
- Wird in vielen Bereichen neben der Softwareentwicklung eingesetzt

- Rollen
  - Product Owner: Produktvision bestimmt Prioriäten
  - Entwicklungsteam: Umsetzen dieser Vision
  - Scrum Master: Sichert Autonomie des Entwicklungsteams, Löst Probleme im Team

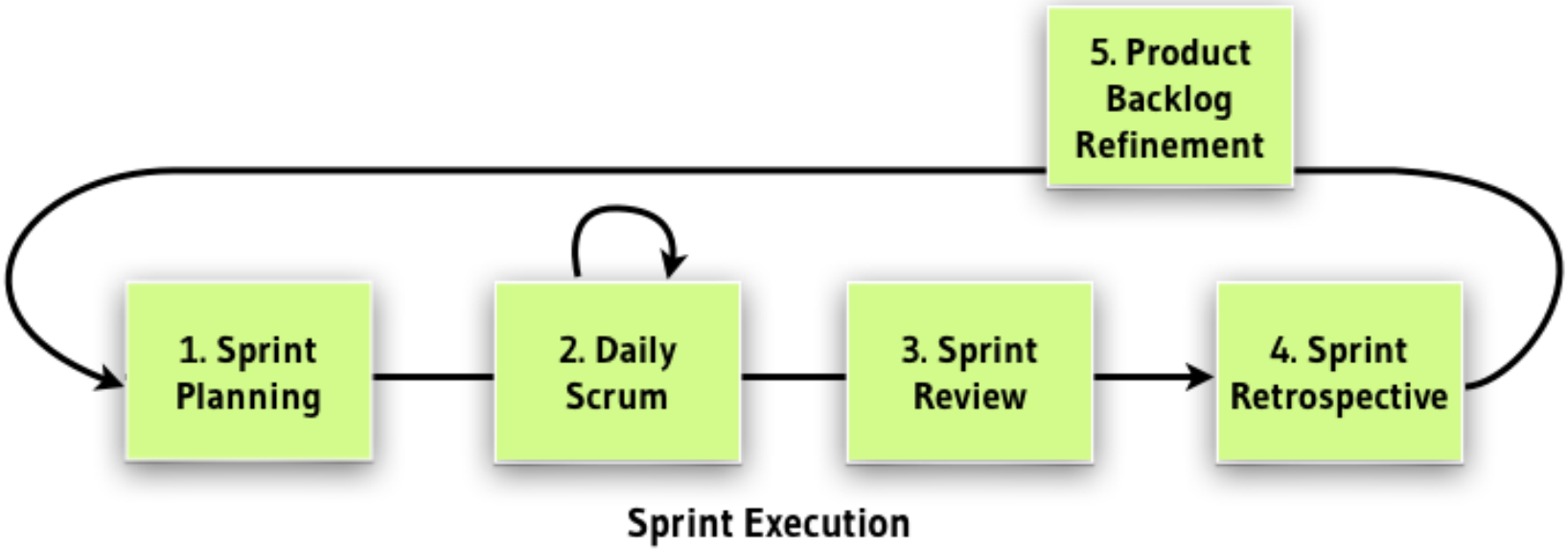
Scrum – Sprint und Product Backlog

- Sprint
  - Circa ein-monatige Entwicklungsphase
  - Nach jedem Sprint liegt ein potenziell fertiges Produkt vor – das Inkrement
- Product Backlog – inkrementelle Planung
  - Definiert alle Anforderungen in priorisierter Reihenfolge
  - Refinement:
    - Jederzeit Hinzufügen, Weglassen, Ändern von Details und Anforderungen

Scrum


kann kurz & informell sein
24 hours
Daily Scrum Meeting
Sprint
30 days
Backlog tasks expanded by team
Sprint Backlog
Product Backlog
As prioritized by Product Owner
Potentially Shippable Product Increment
Source: Adapted from Agile Software Development with Scrum by Ken Schwaber and Mike Beedle.

Scrum – Sprint Bestandteile



[http://agile4ux.com/wp-content/uploads/2014/02/Scrum-Activities.png]

Scrum – Sprint Bestandteile

- Sprint Planning
  - Was kann im Sprint entwickelt werden?
  - Wie wird im Sprint entwickelt?
- Daily Scrum tägliches Meeting
  - Überblick über aktuellen Stand
- Sprint Review stimmt Produkt mit Kundenwünschen überein?
  - Überprüfung des Inkrements mit Stakeholdern
- Sprint Retrospektive
  - Überprüfung der Zusammenarbeit

Zusammenfassung

- Entwicklungsprozesse stellen Kernelemente in der Softwaretechnik dar
- Ordnen die Basisaktivitäten im Lebenszyklus eine Softwareprodukts
- Unterscheidung zwischen
  - Plan-basierten Entwicklungsmethoden
  - Agilen Entwicklungsmethoden

- Welche Entwicklungsmethode die passende ist hängt vom Kontext bzw. Rahmenbedingungen ab!