

# Introduction to Software Engineering

Felix Leidl

26. Februar 2024

## Inhaltsverzeichnis

<b>Software processes</b>	<b>3</b>
Software specification . . . . .	3
Requirements election and analysis . . . . .	3
Requirements specification . . . . .	3
Requirements validation . . . . .	3
Software design and implementation . . . . .	3
Architectural design . . . . .	3
Database design . . . . .	3
Interface design . . . . .	3
Component selection and design . . . . .	4
Software verification and validation . . . . .	4
Component testing . . . . .	4
System testing . . . . .	4
Customer testing . . . . .	4
Software evolution . . . . .	4
Change anticipation . . . . .	4
Change tolerance . . . . .	4
Software development life circle . . . . .	5
Life cycle phases . . . . .	5
Software Process Models . . . . .	5
Wasserfallmodell . . . . .	6
Improved Waterfall model . . . . .	7
V-Modell . . . . .	8
Wiederverwendungsansatz . . . . .	10
Agiles Modell . . . . .	11
Change Management . . . . .	11
Prototyp . . . . .	11
Schrittweise Veröffentlichung . . . . .	12
<b>Agile software development</b>	<b>12</b>
<b>Requirements engineering</b>	<b>13</b>

System modeling	13
Architectural design	13
Design patterns	13
Implementation	13
Software testing	13
Software evolution	13
Software project management	13
Software engineering in machine learning	13

## **Software processes**

### **Software specification**

#### **Requirements election and analysis**

- Beobachtung des existierenden Systems
- Absprache mit Nutzern und Entwicklern
- Anforderungsanalyse
- Entwicklung von Modellen und Prototypen

#### **Requirements specification**

- Anforderungen formulieren und dokumentieren
- Nutzeranforderungen (abstrakt)
- Systemanforderungen (detailliert)

#### **Requirements validation**

- Umsetzbarkeit
- Konsistenz
- Vollständigkeit
- Fehlerkorrektur

## **Software design and implementation**

### **Architectural design**

- Systemstruktur
- Hauptsächliche Strukturen und Beziehungen
- Vertrieb

### **Database design**

- Datenstrukturen
- Darstellung in Datenbanken

### **Interface design**

- Eindeutige Interface-Spezifikationen
- Kommunikation zwischen Komponenten, ohne Kenntnis der Implementation

### **Component selection and design**

- Suche nach wiederverwendbaren Komponenten
- Definiere Veränderungen bei wiederverwendeten Komponenten
- Entwerfe neue Komponenten

### **Software verification and validation**

#### **Component testing**

- Komponenten durch Entwickler testen
- Individuelle Tests, ohne andere Komponenten

#### **System testing**

- Komplettes System ist getestet
- Fehler von unvorhergesehenen Verwendungen und Interfaces sind behoben
- Beweise, dass das System die Anforderungen erfüllt

#### **Customer testing**

- Letzte Hürde vor Veröffentlichung
- System ist von Nutzer mit echten Daten verwendet worden
- Anforderungsprobleme müssen behoben werden

### **Software evolution**

Es gibt zwei Wege mit Veränderung umzugehen:

#### **Change anticipation**

- Mögliche Veränderungen vorhersehen
- Neustart minimieren
- z.B. erst Prototyp erstellen, dann das ganze Produkt

#### **Change tolerance**

- Design berücksichtigt Veränderungen am System
- Normalerweise durch schrittweise Entwicklung
- Ein kleiner Schritt ist genug um eine Veränderung anzunehmen

## Software development life circle

### Life cycle phases

1. Initialisierung, Konzept entwickeln, vorläufige Planung, Anforderungsanalyse (→ Spezifikation)
2. Design: High-level & Low-level Design
3. Implementation
4. Validierung & Verifikation
5. Vertrieb: Veröffentlichung der Anwendung
6. Erhaltung (→ Evolution)
7. Beginne von vorne
8. Bis Absetzung: Planen der Entfernung der Software (aufräumen & archivieren)

## Software Process Models

Ein Prozess-Modell ist eine abstrakte Repräsentation der Aktivitäten während des Softwareentwicklungsprozesses um:

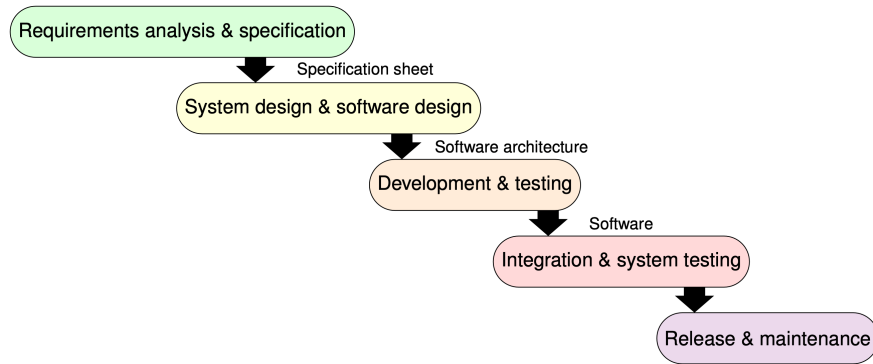
- Abläufe zu definieren
- die Ablaufordnung zu spezifizieren
- Phasen zu determinieren: Abläufe, Ziele, Rollen und Methoden

Die Nutzung von Prozess-Modellen führt zu:

- einer Richtlinie für die Systementwicklung
- einer einheitlichen Ansicht gegenüber logischer und temporärer Planung
- besserer Planung
- Unabhängigkeit von einzelnen Personen
- möglichen Zertifikaten
- früherer Erkennung von Fehlern durch Tests

## Wasserfallmodell

Tabelle 1: Waterfall model



**Requirements analyses & specification:** Projektmanagement beginnt, Probleme und Spezifikationen werden zusammengestellt, Anforderungen definiert und dokumentiert

**System & softwaredesign:** Entwürfe, Modelle und die Softwarearchitektur werden entwickelt

**Development & testing:** Software entwickeln und durch Unit-Tests verifizieren

**integration & systemtests:** Software Komponenten kombinieren und das Gesamtsystem testen

**Release & maintenance:** System installieren, Fehler korrigieren, Software an Altern hindern, neue Anforderungen bearbeiten

### Pros:

- Linearer Prozess
- Intuitiv
- Einfach verständlich
- Top-Down
- Planbar
- Nicht-Unterbrechbar

### Cons:

- Feste Phasen
- Frühe Festlegung
- Keine Wiederholung
- Kein Einbeziehen neuer Anforderungen
- Oft unpraktisch

### Verwendung:

- Anforderungen sind einfach zu definieren und ändern sich nicht
- Projekt, Budget und Prozess sind vorhersagbar
- Strikter Prozess ist notwendig

## Improved Waterfall model

Tabelle 2: Iterative Waterfall model

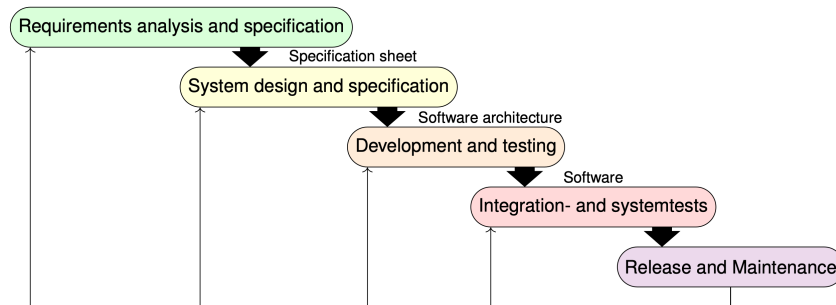
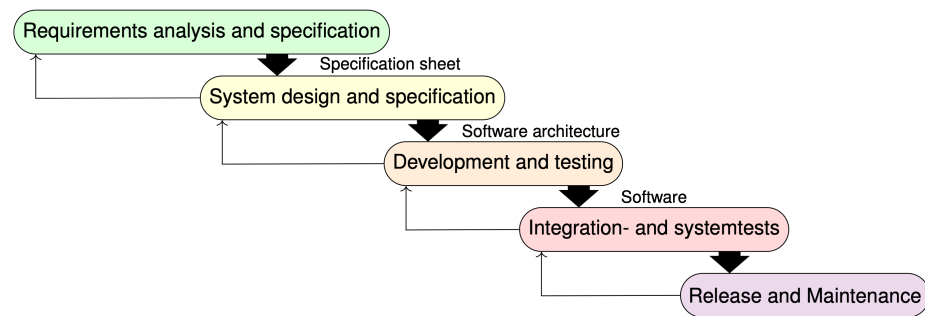


Tabelle 3: Incremental Waterfall model



### Pros:

- Linearer Prozess
- Intuitiv
- Einfach zu Verstehen
- Top-Down
- Planbar
- Nicht-Unterbrechbar
- Wiederholbar

### Cons:

- Gefixte Phasen
- Frühe Festlegung, aber neue Anforderungen können integriert werden
- Veränderte Anforderungen können zu hohen Kosten führen
- Struktur tendiert abzubauen

### Verwendung:

- Anforderungen sind einfach definiert

- Nur kleine Änderungen können erscheinen und sind im Budget mit eingerechnet
- Projekt, Budget und Prozess sind vorhersagbar
- Strikter, aber leicht flexibler Prozess ist notwendig

## V-Modell

Tabelle 4: V-model

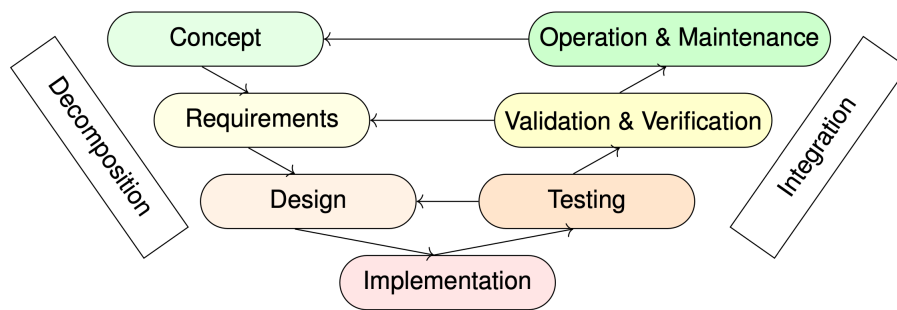
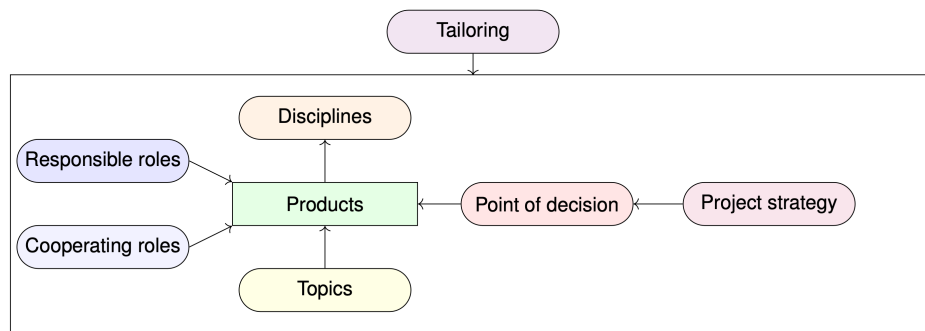
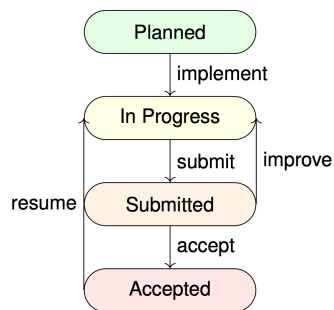




Tabelle 5: V-model XT



Jedes Produkt geht durch vordefinierte Zustände



**Pros:**

- Bei großen und komplexen Systemen anwendbar
- detaillierte Spezifikationen, Rollendefinitionen und Ergebnisstrukturen
- Qualitätsorientiert

**Cons:**

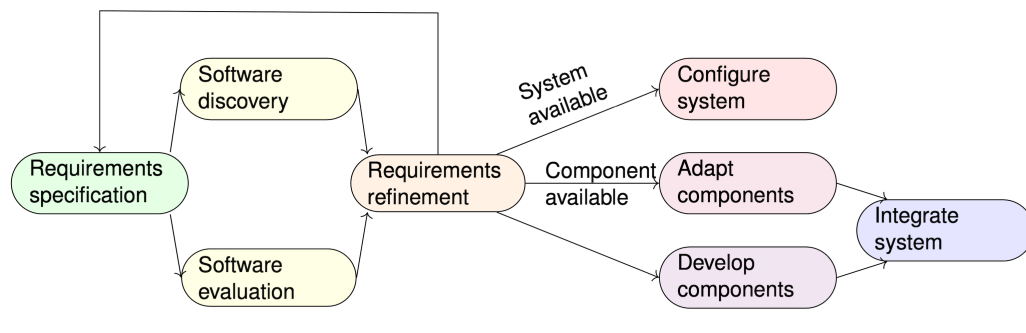
- Großer Overhead bei kleinen Projekten
- Testphase beginnt relativ spät
- Strikte Phasen
- Teilnehmer müssen geschult sein, um Modell zu folgen

**Verwendung:**

- Softwareentwicklung in Behörden
- Sicherheitsrelevante Projekte

## Wiederverwendungsansatz

Tabelle 6: Reuse-oriented approach



### Pros:

- Reduziert Kosten und Risiken
- Schnellere Verteilung

### Cons:

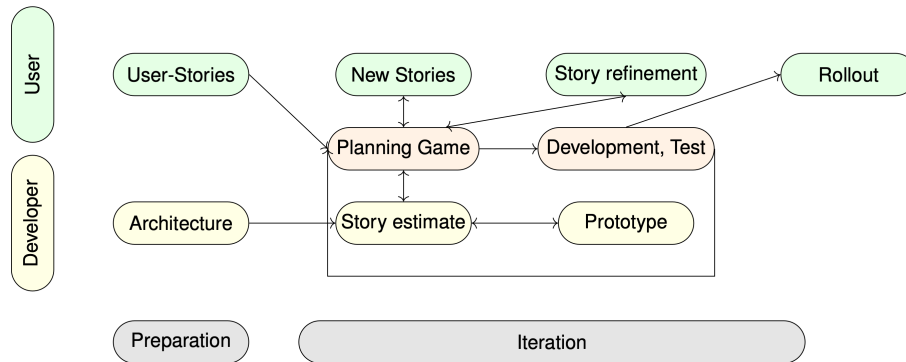
- Kompromisse in den Anforderungen
- System könnte echte Nutzerbedürfnisse nicht erfüllen
- Kein oder limitierte Kontrolle über Systemevolution

### Verwendung:

- Webanwendungen
- Collection of objects or packages
- Konfigurierbare stand-alone Softwaresysteme

## Agiles Modell

Tabelle 7: Agile model



### Pros:

- Begrenzter bürokratischer Aufwand
- Flexible Rollen
- so wenig Dokumentation wie nötig
- besseres Kosten/Nutzen Verhältnis
- Bessere Code-Qualität

### Cons:

- Ganzes Team muss den Regeln folgen
- Projektergebnis nicht vorhersehbar

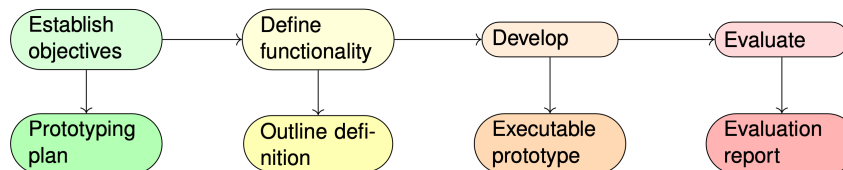
### Verwendung:

- Große, komplexe sowie kleine Systeme
- Projekte die Prototypen erfordern

## Change Management

### Prototyp

Tabelle 8: Prototyping



### Pros:

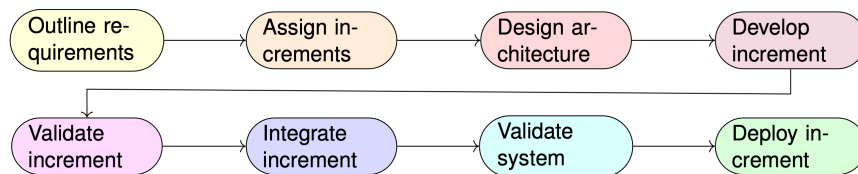
- Kunde gibt Priorität der Anwendung vor
- Software kann sofort verwendet werden
- Kunde erlangt Erfahrung mit dem System
- Kunde kann Anforderungen für spätere Schritte abgeben
- Veränderungen sind einfach umzusetzen
- Die wichtigsten Systemteile werden am häufigsten getestet

### Cons:

- Softwarebasis kann ohne detaillierte Anforderungen nicht identifiziert werden
- Kann mit organisatorischen Strukturen in Konflikt geraten (z.B. Verträge)
- Unbrauchbar um existierende Systeme zu ersetzen

### Schrittweise Veröffentlichung

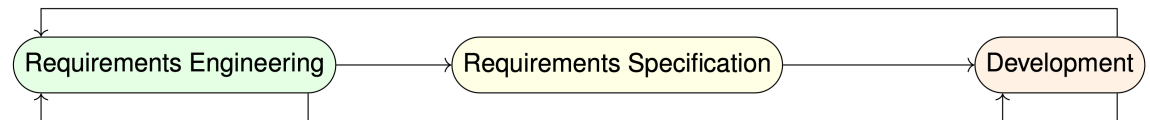
Tabelle 9: Incremental delivery



### Agile software development

Tabelle 10: diff. between plan-driven and agile

Plan-driven:



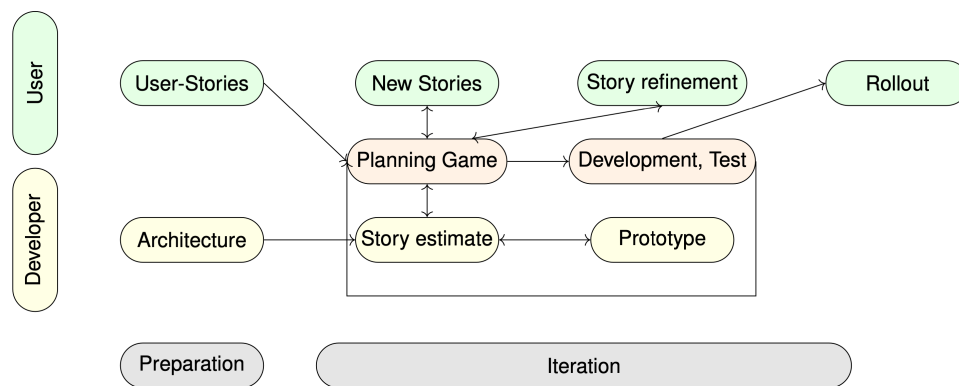
Agile:



### Agile manifesto:

1. Individuen und Interaktionen über Prozessen und Werkzeugen
2. Funktionierende Software über akribischer Dokumentation
3. Zusammenarbeit mit dem Kunden über Vertragsverhandlungen
4. Auf Veränderungen eingehen über Plan folgen

Tabelle 11: Generic model



Requirements engineering

System modeling

Architectural design

Design patterns

Implementation

Software testing

Software evolution

Software project management

Software engineering in machine learning