# Software Engineering

Lernskript

Dozent:
Prof. Dr. Stefan Kramer

LaTeX von:
Sven Bamberger

Zuletzt Aktualisiert:
23. Februar 2014

Dieses Skript wurde erstellt, um sich besser auf die Klausur vorzubereiten.

Dieses Dokument garantiert weder Richtigkeit noch Vollständigkeit, da es aus Mitschriften und Vorlesungsfolien gefertigt wurde und dabei immer Fehler entstehen können. Falls ein Fehler enthalten ist, bitte melden oder selbst korrigieren und neu hoch laden.

# Inhaltsverzeichnis

# 1 Zusammenfassung - Lernziele

- **Software Engineering ist eine Ingenieursdisziplin, die sich mit allen Aspekten der Softwareproduktion beschäftigt**
- High-level activities specifications, Entwicklung, Validierung und Evolution sind Teil aller Softwareprozesse.
- Prozesse:
  - Wasserfallmodell
  - V-Modell
  - Boehm's spiral model
  - RUP (modernes generisches Prozessmodell)
  - agile Methoden (Scrum, XP, ...),...
- Es gibt **viele verschiedene Systemformen**, ihre Entwicklung erfordert eigene Software Engineering-Tools und Techniken
- **Requirements engineering** ist der Prozess der Entwicklung einer Software-Spezifikation
  - Kernpunkte: User- und System Requirements, Funktionale und nicht funktionale Anforderungen, Qualitätseigenschaften der Anforderungen
- Design: **high-level design** (architectural design) und **low-level design**
  - Pattersn als Weg, bekannte, erfolgreiche Designstrategien zu übertragen
- **Verfication und Validation:** Inspecktion (*code reviews*) und Testen (viele Verschiedene Formen des Testings:
  - defect testing
  - validation testing
  - unit testing
  - component testing
  - system testing
  - regression testing
  - ...
- Project Management
  - Einhaltung des Zeitplans und des Budgets
  - **Risikomanagement** gilt als einer der wichtigsten Punkte, Hauptwerkzeug: Notfallplan
  - Personalmangement: Bedürfnishierachie, Persönlichkeitstypen
- Project planning
  - Milestones, deliverables, Gantt (or activity bar) chart
  - **Kostenabschätzung COCOMO2**

- Quality mangement
  - **Prüfplan, ISO 9001-Standart, software metrics**
- **CMMI-Modell für Prozessoptimierung**

# 2 Introduction to Software Engineering

## 2.1 Was versteht man unter Software-Engineering?

**Software Engineering** ist eine Ingenierusdisziplin, die sich mit allen Aspekten der Softwareherstellung beschäftigt.

## 2.2 Technische Disziplin:

- Techniker wenden Methoden, Werkzeuge und Theorien an um Dinge zum Laufen zu bringen
- Techniker erkennen an, dass sie mit oganisatorischen und finanziellen Beschränkungen arbeiten müssen

## 2.3 Alle Aspekte der Softwareherstellung:

- SE beschäftigt sich nicht nur mit technischen Prozessen der Softwareentwicklung
- auch Projektverwaltung und Entwicklung von Werkzeugen, Methoden und Theorien, welche die Sofwareherstellung unterstützen

## 2.4 Softwareprozess (grundlegende Aktivitäten von SE):

- Softwarespezifikation
  - Kunden und Entwickler definieren die zu produzierende Software und die Rahmenbedingung fr ihren Einsatz
- Softwareentwicklung
  - Software wird entworfen und programmiert
- Softwarevalidierung (Softwarebeurteilung):
  - Software wird überprüft um sicherzustellen, dass sie den Kundenanforderungen entspricht
- Softwareevolution
  - Software wird weiterentwickelt, damit sie die sich ändernden Bedürfnisse der Kunden und des Marktes wiederspiegelt

## 2.5 Software-Produkte

- **Generisch:** Stand-alone-Systeme, dei vermakted und an jeden Interessenten verkauft werden
  - Beispielsweise: Software für den PC, Datenbanken, Textverarbeitungsprogramme
- **Produktspezifikation:**

  - Anforderungen werden vom Entwickler definiert, Entscheidungen über Änderungen werden vom Entwickler getroffen.

- **Kundenindividuell:** Software die im Auftrag eines bestimmten Kunden hergestellt werden
  - Biespielsweise: Steuerungssysteme für elektronische Geräte, System zur Unterstützung eines bestimmten Geschäftprozesses

- **Produktspezifkation:**
  - Anforderungen werden vom Kunden definiert, Entscheidungen über notwendige Änderungen der Software werden vom Kunden getroffen

## 2.6 Unterschiede dieser Softwaretypen:

- Allgemeine Produkte:
  - Unternehmen, das Software entwickelt, bestimmt die Spezifikationen der Software
- Angepasste Produkte:
  - Spezifikation wird von dem Kunden entwickelt und kontrolliert, dass das System kauft

## 2.7 Viele unterschiedliche Anwendungsarten:

- Eigenständige (stand-alone) Anwendungen:
  - Softwware läuft lokal
  - Beinhaltet alle notwendigen Funktionen um ohne Netzwerkverbindung zu laufen
  - z.B. Fotoanwendungen ohne Netzwerkzugriff
- Interaktive transaktionsbasierte Anwendungen:
  - laufen auf externen PC's und werden vom User aufgerufen
  - Webapplikationen über Cloud-Zugriff
- Eingebettete Steuerungssysteme:
  - es gibt daon mehr als von allen andern Anwendungsarten
  - Software für ABS im Auto
  - Kontrollieren die Hardware
- Stapelverarbeitende Systeme (Batch-Verarbeitungssysteme)
  - Systeme die große Datenmengen verarbeiten (z.B. Business-Systeme)
  - Abrechnung von regelmäßigen Zahlungen (Telefonrechnung)
- Unterhaltungssysteme
  - zum persönlichen Gebrauch
  - z.B. Spiele
- Systeme für die Modellierung und Simulation
  - Systeme, die von Wissenschaftlern und Ingenieuren entwickelt wurden
  - z.B. physikalische Prozesse simulieren

- Datenerfassungssysteme
  - System, welches Daten unter extremen Bedingungen mit Hilfe von Sensoren erfassen und weiterleiten
- Systeme von Systemen
  - Zusammensetzung von vielen Softwaresystemen

## 2.8 Software Engineering FAQ

1. Software Definition
   - Computerprogramme und dazugehörige Dokumentationen können für bestimmte
2. Attribute guter Software
   - Liefert geforderte Funktionalität und Leistung, ist wartbar, zuverlässig, effizient und benutzbar
3. Software Engineering Definition
   - Ingenieursdiziplin, die sich mit allen Aspekten der Softwareproduktion befasst.
4. Software Enigneering: Grundsätzliche Tätigkeiten
   - Software-Spezifikation, Entwicklung, Validierung, Evolution
5. Unterschied Software Engineering ↔ Systementwicklung
   - Informatik: Theorie und Grundlagen
   - Softwareentwicklung: Praktisches Umsetzung der Entwicklung und Auslieferung nützlicher Software
6. Unterschied Software Engineering ↔ Systementwicklung
   - Systementwicklung beinhaltet alle Aspekte computerbasierter Entwicklung, Hardware-, Software-, Prozessentwicklung
7. Zentrale Herausforderungen des Software Engineerings
   - zunehmende Vielfalt, Forderung verkürzter Lieferzeiten, Entwicklung vertrauenswürdiger Software
8. Kosten des Software Engineerings
   - 60% Entwicklung, 40% Testing
9. Beste Techniken und Methoden
   - passende Techniken für unterschiedliche Systemarten
10. Einfluss des WWW

## 2.9 Wesentliche Merkmale guter Software:

- Wartbarkeit
  - kritisches Merkmal
  - Software sollte so geschrieben werden, dass sie **weiterentwickelt werden kann**, um sich verändernden Kundenbedüfnissen Rechnung zu tragen

  - Softwareanpassungen sind eine unvermeidliche Anforderung einer sich verändernden Geschäftsumgebung

- Verlässlichkeit (dependability) und Informationssicherheit (security):
  - Verlässliche Software sollte keinen körperlichen oder wirtschaftlichen Schaden verursachen, falls das System ausfällt
  - Böswillige Benutzer sollten nicht in der Lage sein, auf das System zuzugreifen oder es zu beschädigen

- Effizienz (efficiency):
  - Software sollte nicht verschwenderisch mit Systemressourcen wie Speicher und Prozessortkapazität umgehen
  - Effizien umfasst somit Reaktionszeit, Verarbeitungszeit, Speichernutzung usw.

- Akzeptanz (acceptability)
  - Software muss von den Benutzern akzeptiert werden, für die sie entwickelt wurde. Das bedeutet, dass sie verständlich, nützlich und kompatibel mit anderen Systemen sein müssen, die diese Benutzer verwenden.

## 2.10 Grundsätze des Software Engineering:

- Einsatz eines kontrolliertern, vereinbarten Entwicklungsprozesses
- Zuverlässigkeit und Leistungsfähigkeit wichtig
- Verständnis und Kontrolle der Software-Spezifikation und der Anforderungenen sind wichtig
- Wiederverwendung bereits vorhandener Software sollte der Vorzug vor der Entwicklung neuer Software vorgezogen werden.

## 2.11 Key points

1. Softwareentwicklung ist eine Ingenieursdisziplin, die sich mit allen Aspekten der Softwareproduktion beschäftigt.
2. Wesentliche Attribute eines Software-Produktes sind Wartbarkeit, Zuverlässigkeit und Sicherheit, Effizienz und Abnahmefähigkeit.
3. high-level-Aktiviäten der Spezifikation, Entwicklung, Validierung und Evolution sind Teil aller Softwareprozesse.
4. Grundlegende Ideen der Softwareentwicklung sind universell andwendbar auf alle Formen von Systementwicklung.
5. Jede Art von System erfordert passende Entwicklungs-Tools.
6. Die grundlegende Ideen der Softwareentwicklung sind anwendbar für alle Formen von Sostwaresystemen.

# 3 Softwareprozesse

Softwareprozess ist eine Menge von zusammengehörigen Aktivitäten, die zur Produktion eines Softwareprodukts führen.

## 3.1 Vier Aktivitäten die grundlegend für SE sind:

- Softwarespezifikation:
  - Beschreibung was das System macht
- Softwareentwurf und -implementierung
  - Definieren des Systemaufbaus
  - Implementierung des Systems
- Softwarevalidierung
  - Sind Kundenwünsche erfüllt
- Softwareevolution
  - Weiterentwicklung und eingehen auf veränderte Bedürfnisse des Kunden

## 3.2 Plangesteurte und agile Prozesse:

- Plangesteurte Prozesse:
  - Verfahren bei denen alle Prozessaktivitäten im Voraus geplant werden
  - Fortschritte werden an dem Plan gemessen
- Pro:
  - durch frühzeitige Planung werden Probleme und Abhängigkeiten entdeckt (z.B. Personalbedarf)
- Contra:
  - viele frühe Entscheidungen müssen korrigiert werden (z.B. wegen Änderungen in der Umgebung)
- Agile Prozesse:
  - Planen ist inkrementell und es ist einfacher Prozesse zu verändern
- Es muss eine Balance zwischen beiden Prozessen gefunden werden

## 3.3 Vorgehensmodelle (Softwareprozessmodelle):

- Wasserfallmodell
- Inkrementelle Entwicklung
- Wiederverwendungsorientierte SE

### 3.3.1 Wasserfallmodell:

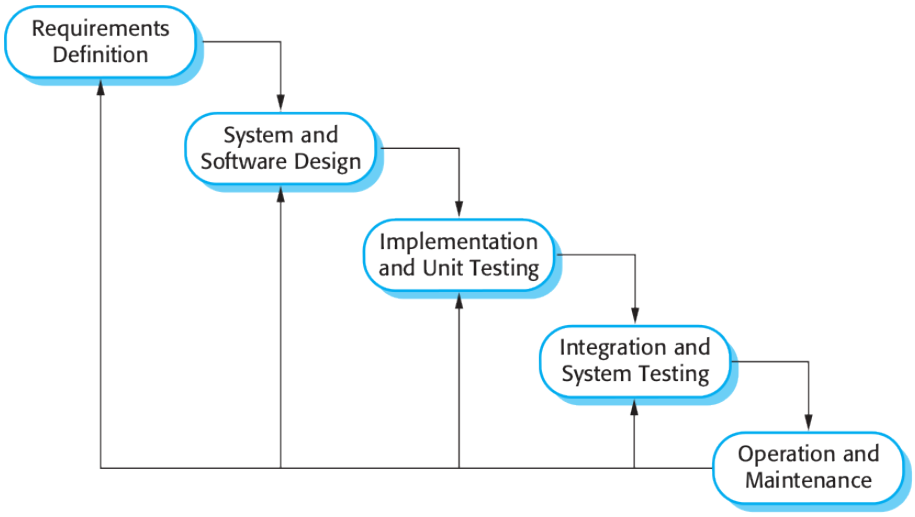


Abbildung 3.1: Wasserfallmodell: Plangesteuerter Prozess

- Modell stellt die grundlegende Prozessabläufe (Spezifikation, Entwicklung, Validierung und Evolution) als eigene Phase des Prozesses dar
- Phasen
  1. Analyse und Definition der Anforderungen dienen als Systemspezifikation
  2. System- und Softwareentwurf:
     - Übergeordnete Systemarchitektur wird festgelegt
     - Erkennen und Beschreiben der grundlegenden abstrakten Softwaresysteme
  3. Implementierung und Modultests
  4. Integration und Systemtest
     - System wird als Ganzes getestet zur Siherheitsstellung, dass die Softwareanforderungen erfüllt sind.
  5. Betrieb und Wartung
- Hauptproblem ist die starre Aufteilung des Projekts in verschiedene Phasen
- Man kann nicht gut auf neue Anforderungen des Kunden reagieren
- gute „Wahl“, wenn Anforderungen gut durchdacht sind und wenn keine Änderungen zu erwarten sind
- geeignet für große Systeme

### 3.3.2 Inkrementelle Entwicklung

- Spezifikation, Entwicklung und Validierung sind verschachtelt. Kann sowohl plangesteuertals auch agil sein.
- System wird als eine Folge von Versionen (Inkrementen) entwickelt, wobei jede Version neue Funktionalität zu der vorherigen hinzufügt

- Spezifikation, Entwicklung und Validierung werden gleichzeitig ausgeführt

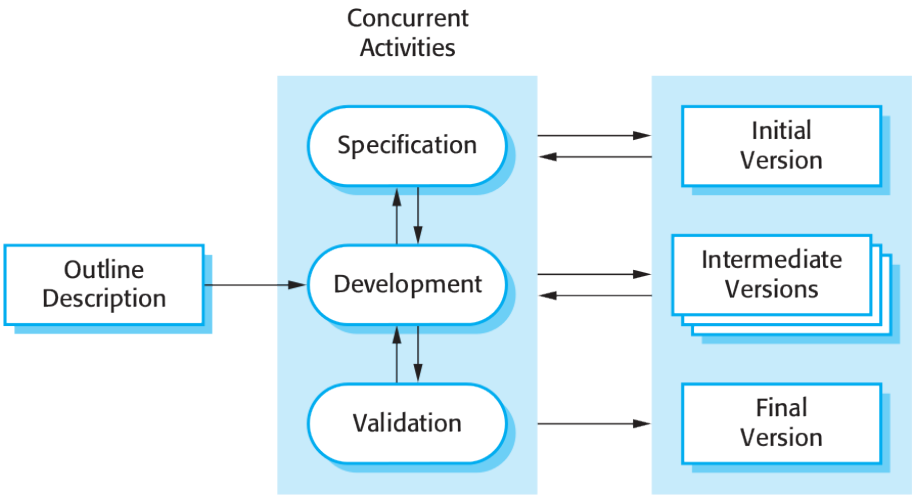


Abbildung 3.2: Inkrementelle Entwicklung

**Vorteile inkrementelle Entwicklung (gegenüber Wasserfallmodell):**

- Kosten für Kundenanforderungsänderungen werden reduziert
- Es ist einfacher Kundenrückmeldungen zu bekommen
- Schnellere Auslieferung an den Kunden von verwendungsfähiger Software

**Nachteile inkrementelle Entwicklung**

- Prozess ist nicht sichtbar
  - Fortschritt nicht messbar
- Systemstruktur wird tendenziell schwächer, wenn neue Inkremente hinzugefügt werden

### 3.3.3 Inkrementelle Lieferung

- Entwicklung und Auslieferung werden in mehrere Schritte aufgeteilt, jeder Schritt liefert einen Teil der geforderten Funktionalitäten
- Den Benutzeranforderungen werden Prioritäten zugewiesen, die Anforderungen mit höherer Priorität werden in früheren Schritten bearbeitet
- während der Bearbeitung eines Schrittes werden die betreffenden Anforderungen nicht mehr verändert

**Vorteile:**

- Schrittweise gelieferte Systemfunktionalität
- Frühe Schritte dienen als Vorlage, mit der spätere Schritte bestimmt werden
- Geringes Risiko für Fehlschlag der gesamten Software
- die hoch priorisierten Systemdienstleistungen werden oft getestet

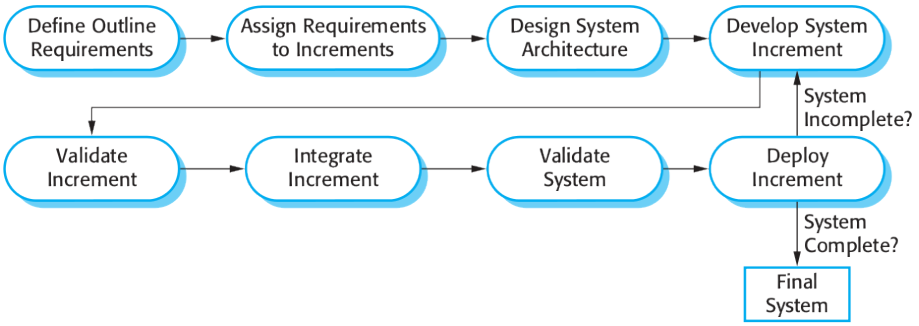


Abbildung 3.3: Inkrementelle Lieferung

**Nachteile:**

- Schwer zu verwalten
- Systemstruktur wird aufgeweicht
- Kompatibilität mit Geschäftsprozessen auf Kundenseite nicht gesichert
- Ersetzen eines bestehenden Systems problematisch → Schritte enthalten weniger Funktionalität

### 3.3.4 Wiederverwendungsorientiertes Software Engineering

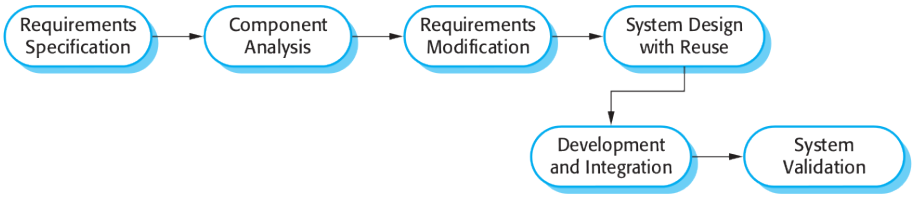


Abbildung 3.4: Reuseoriented software engineering

- beides (Plandriven & agiler Prozess)
- basiert auf Existenz einer beträchtlichen anzahl von wiederverwendbaren Komponenten
- Prozessstufen:
  - Komponenten Analyse
  - Anforderungsmodifikation
  - Systementwurf mit Wiederverwendung
  - Entwicklung und Integration
- meist aus allen Modellen zusammengesetzt

### 3.3.5 Typen der Software-Komponenten:

- Stand-Alone-Software (COTS Commercial-off-the-shelf):
  wurde für den Einsatz in einer bestimmten Umgebung Entwickelt

- Sammlung von Objekten, die als Paket mit einem Komponentenframework wie .NET oder J2EE integriert und entwickelt wurden
- Web-Services:
  wurden je nach Servicestandard entwickelt und für Remote-Aufrufe verfügbar gemacht

## 3.4 Prozessaktivitäten:

### 3.4.1 Softwarespezifikation

Prozess zum Verstehen und Definieren der vom System verlangten Funktion sowie der Beschränkungen, denen der Betrieb und die Entwicklung des Systems unterliegen.
**Requirements Engineering**

1. Anforderungsanalyse besteht aus
2. **Durchführbarkeitsstudie:**
   - Ist es aus technischer und finanzierler Sicht möglich das System zu entwickeln?
3. **Erhebung und Analyse der Anforderungen:**
   - Was erwarten die Kunden von dem System
4. **Spezifikation der Anforderungen:**
   - Definieren der Anforderungen im Detail
5. **Validierung der Anforderungen:**
   - Sind die Anforderungen realistisch, konsistent und vollständig?

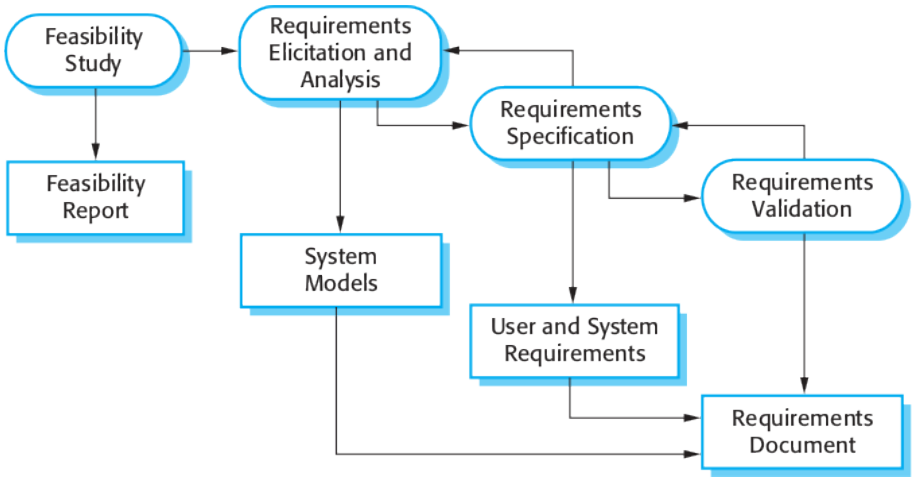


Abbildung 3.5: Softwarespezifikationen

### 3.4.2 Design und Implementierung

Umwandlen der Spezifikationen in ein ausführbares System.

- Software-Design

  - Entwicklung einer Softwarestruktur, die die Spezifikation umsetzt.

- Implementierung
  - Übersetzung der Struktur in ein ausführbares Programm.

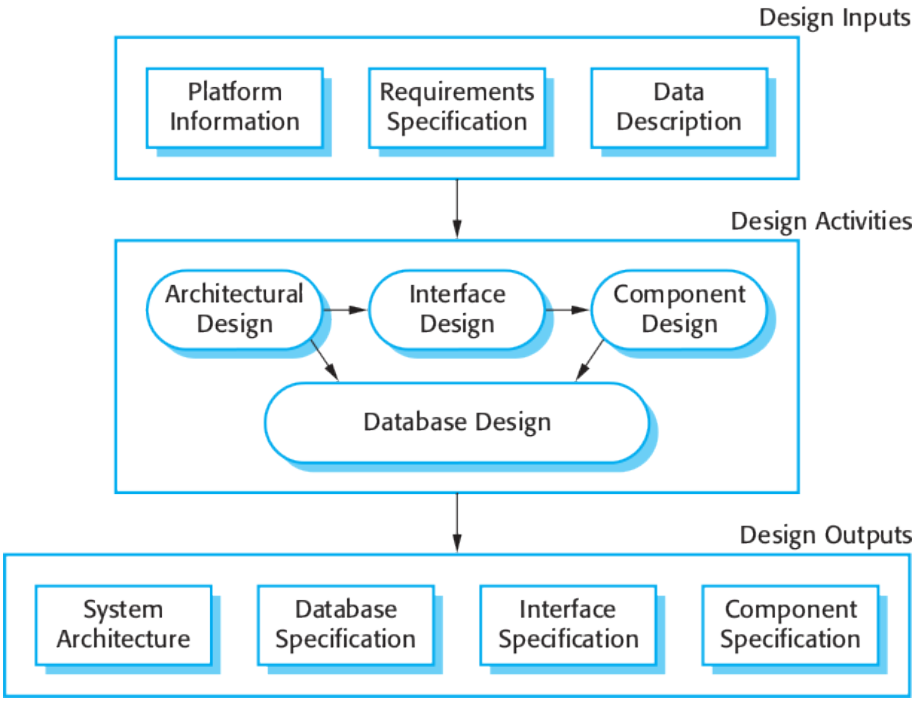


Abbildung 3.6: Design und Implementierung

### 3.4.3 Designaktivitäten

- Architekturentwurf:
  Gesamtstruktur des Systems → Hauptkomponenten (wie Untersysteme und Module), Beziehungen und deren Verteilungen

- Schnittstellenentwurf:
  Definition der Schnittstellen zwischen Systemkomponenten

- Komponentenentwurf:
  Jede Komponente und jedes Design wird genauer betrachtet

- Datenbankentwurf:
  Entwurf der Systemdatenstruktur und deren Darstellung in der Datenbank

### 3.4.4 Software- Validierung

- Verfikation und Validierung soll zeigen, dass ein Systeseiner Spezifikation entspricht und die Anfoerderungen des Kunden erfüllt

- Kontrolle und Bewertungsprozess, Testendes Systems

- Testing: ausführen des Systems mit Testfällen (aus der Spezifikation der Daten, die mit dem System verarbeitet werden sollen)
- Häufigste V&V-Aktivität: Testing

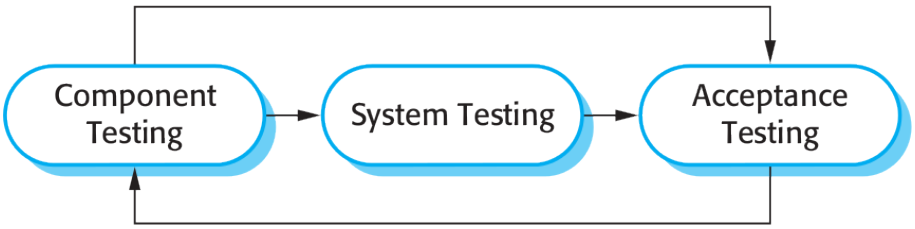


Abbildung 3.7: Testing Zyklus

- Komponenten Testing:
  - Tests von Einzelkomponenten unabhängig von einander
  - Komponenten können Funktionen, Objekte oder kohärente Gruppen von Personen sein
- Systemtest
  - Test des gesamten Systems (entstehender Eigenschaften)
  - Prüfung von emergenten Eigenschaften besonders wichtig
- Abnahmetests:
  - Testen mit Daten des Kunden zur Überprüfung der Anforderungen

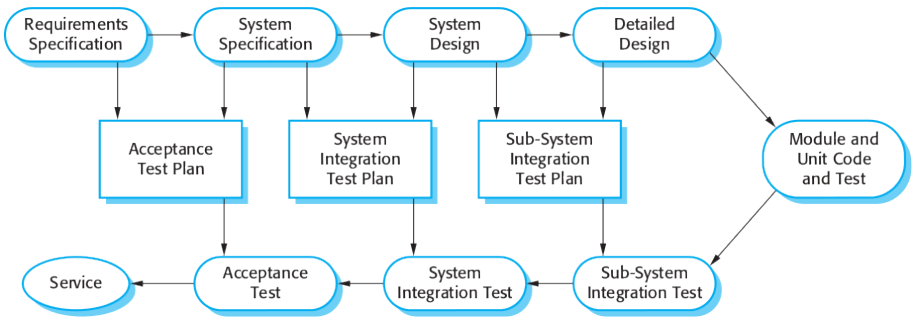


Abbildung 3.8: Testing bei plangesteuertem Software-Prozess

### 3.4.5 Software-Evolution

- Software ist von Natur aus flexibel und kann sich verändern
- Anforderungen änder sich durch wirtschafliche Umstände, die unterstützende Software muss sich mit verändern
- Durch die Wiederverwendung von Komponenten für neue Systeme sind Softwareentwicklung und Evolution nicht mehr vollständig unterscheidbar

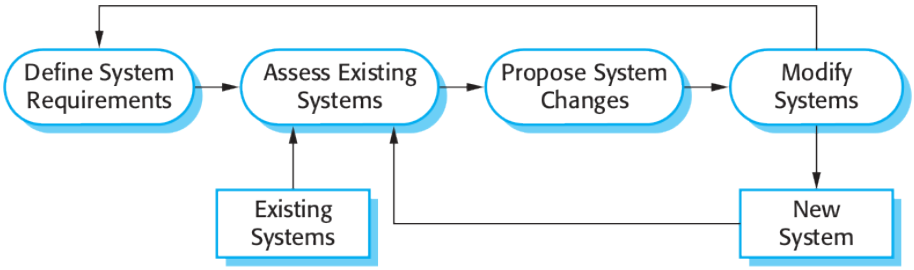


Abbildung 3.9: Software-Evolution

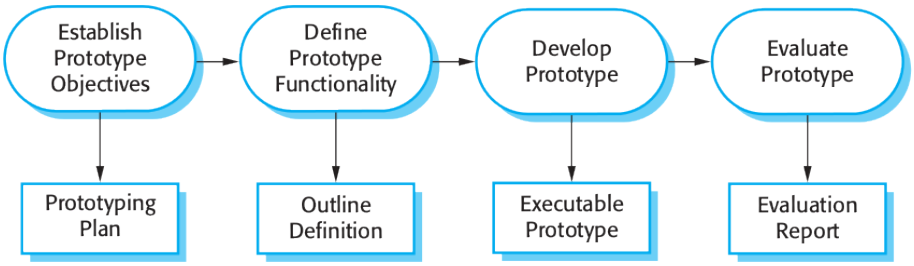


Abbildung 3.10: Prototyping

### 3.4.6 Prototyping

- eine erste Version eines Softwaresystems um Konzepte zu demonstrieren und Erkenntnisse zu gewinnen
- Wird benutzt, wenn:
  - Requirements-Engineering Prozess hilft ein Prototyp bei der Ermittlung und Validierung der Systemanforderungen
  - Beim Systementwurf kann der Prototyp verwendet werden, um einzelne Softwarelösungen zu utnersuchen und den Entwurf der Benutzeroberfläche zu unterstützen

### 3.4.7 Spiralmodell nach Boehm:

- Softwareprozess wird als Spirale dargestellt, anstatt als eine Folge von Aktivitäten
- Jede Windung steht für eine Phase des Prozesses
- keine festen Phasen wie Spzeifikation → Schleifen werden nach Erforderlichkeit gewählt.
- beinhaltet explizite Risikomanagmentaktivitäten
- Jede Windung der Spirale ist in vier Segmente aufgeteilt:
  - Ziele aufstellen
  - Risiken einschätzen und verringern
  - Entwicklungnd Validierung
  - Planung
- Nutzung:
  - sehr einflussreich um Menschen zu helfen über Iterationen in Softwareprozessen und die Eifnührung des risikoorientierten Ansatzes für die Entwicklung nach zu denken

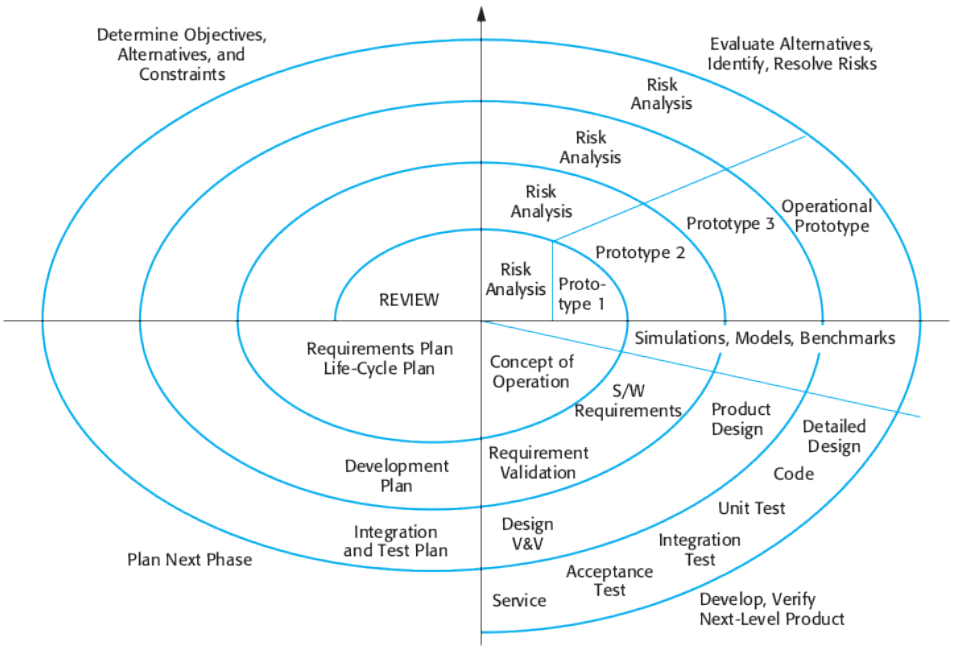


Abbildung 3.11: Prototyping

- Praxis
  - → Modell wird selten verwendet
  - → wird zur Veröffentlichung der praktischen Softwareentwicklung verwendet

### 3.4.8 Rational Unified Process (RUP):

- abgeleitet von der Arbeit an der UML
- Vereinigt Elemente aus allen allgemeinen **Vorgehensmodellen** (Wasserfall, inkrementelle Entwicklung und wiederverwendungsorienteirte SE)
- veranschaulicht empfohlene Vorgehensweisen für Spezifikation und Entwurf
- unterstützt Entwicklung von Prototypen
- RUP wird aus drei Perspektiven beschrieben
  - **dynamische Perspektive:**
    die die Phasen des Modells zeitlich darstellt
  - **statische Perspektive:**
    die die ausgeführten Prozessaktivitäten darstellt
  - **praxisbezogene Perspektive:**
    die die während des Prozesses empfohlene Vorgehensweise vorschlägt
- Sechs grundlegende Vorgehensweisen für das SE zum Einsatz bei der Systementwicklung:
  1. Software iterativ entwickeln:
     - schrittweise Planung des Systems ausgehend von den Prioritäten des Kunden
  2. Anforderungen verwalten:

   - Eindeutige Dokumentation der Kundenanforderungen
   - geänderte Anforderungen verfolgen
3. Komponentenbasierte Architekturen verwenden:
   - Aufteilung der Systemarchitektur in Komponenten (Wiederverwendung)
4. Software visuell modellieren:
   - Verwenden grafischer UML-Modelle für dynamische und statische Sicht
5. Softwarequalität verifizieren:
   - Sicherstellen, dass die Software die Qulaitätsstandards des Unternehmens erfüllen
6. Änderungen der Software steuern (überprüfen):
   - Verwaltung der Softwareveränderungen mithilfe eines Änderungsmanagementsystems

# Abbildungsverzeichnis