Software Engineering

Lernskript

Dozent:
Prof. Dr. Stefan Kramer

LaTeX von:
Sven Bamberger

Zuletzt Aktualisiert:
24. 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:

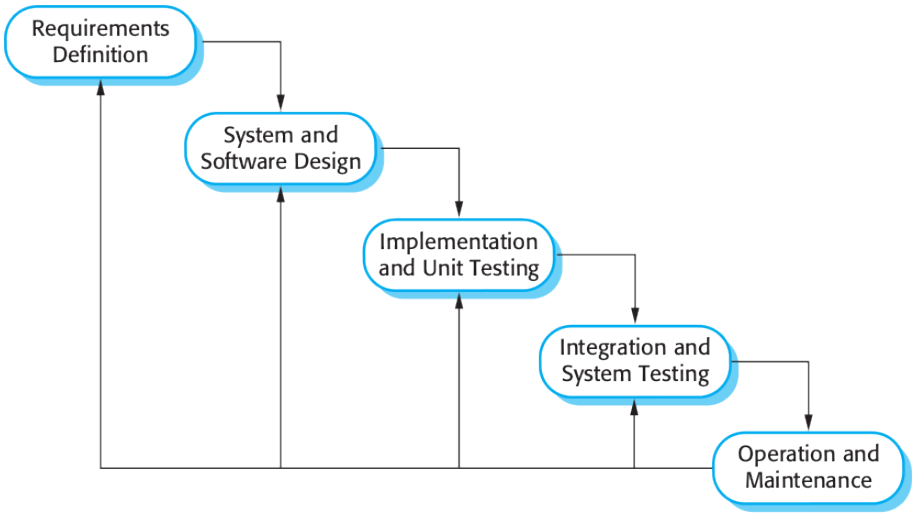

Abbildung 3.1: Wasserfallmodell: Plangesteuerter Prozess

- 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

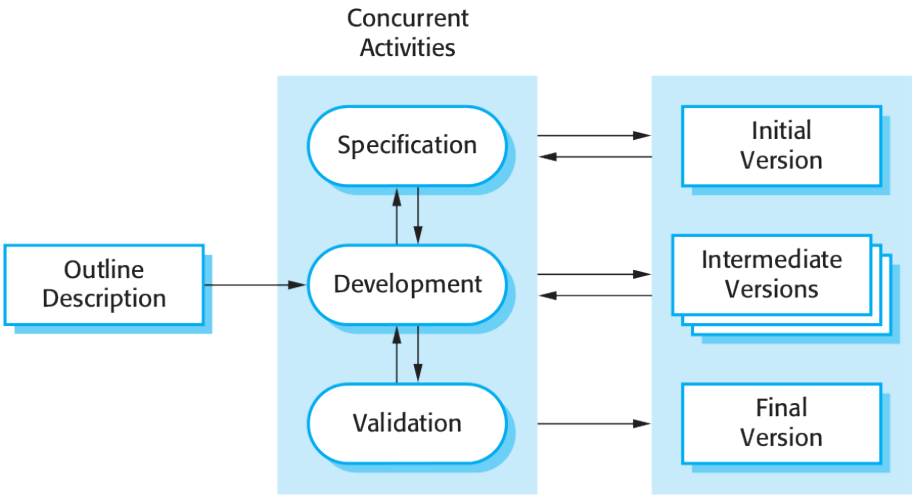

Abbildung 3.2: Inkrementelle Entwicklung

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

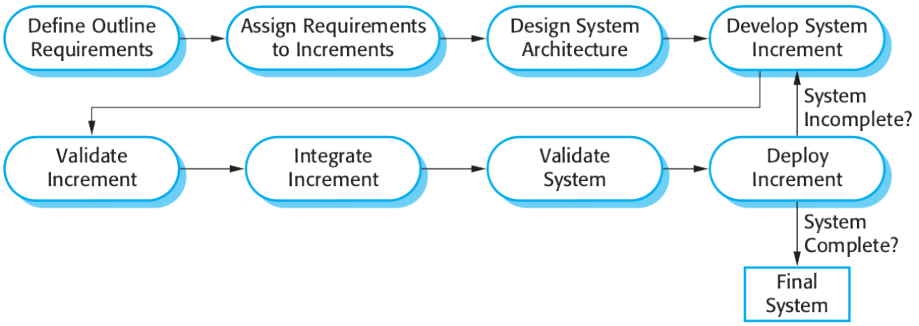

Abbildung 3.3: Inkrementelle Lieferung

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

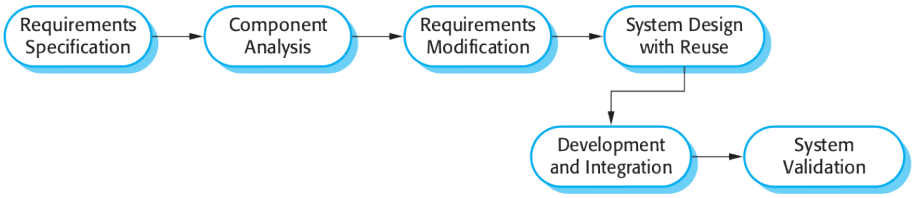

Abbildung 3.4: Reuseoriented software engineering

- 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?

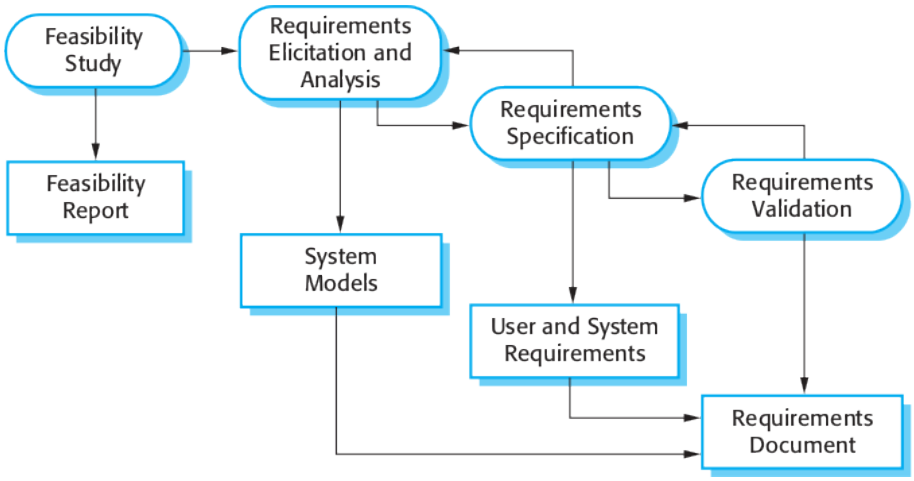

Abbildung 3.5: Softwarespezifikationen

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.

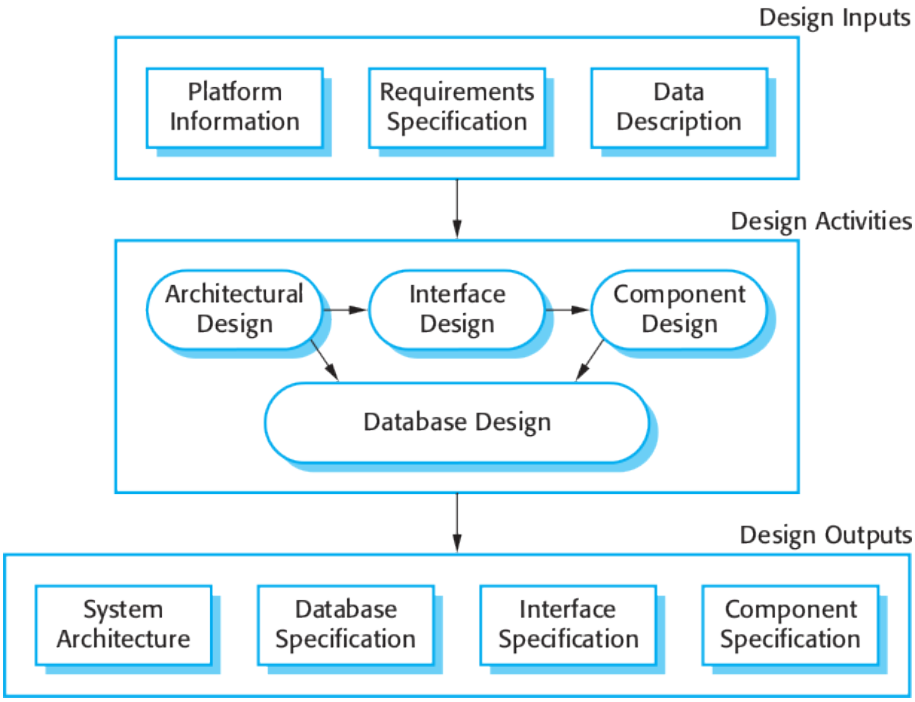

Abbildung 3.6: Design und Implementierung

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

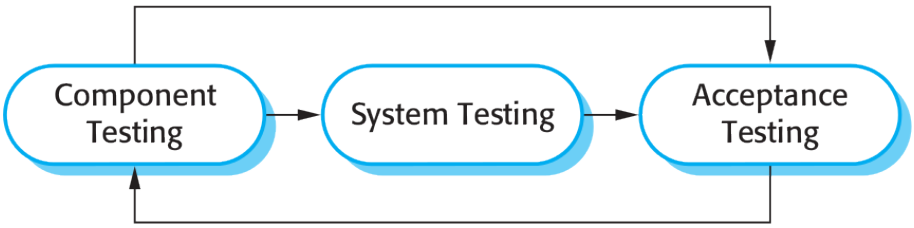

Abbildung 3.7: Testing Zyklus

- 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

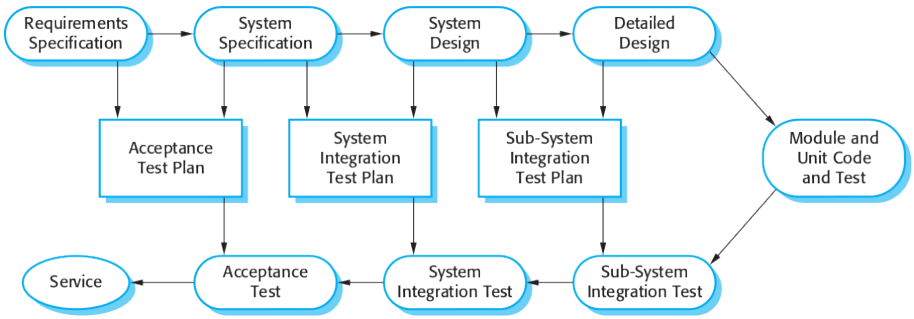

Abbildung 3.8: Testing bei plangesteuertem Software-Prozess

3.4.5 Key Points:

1. Softwareprozesse sind Aktivitäten, die zur Entwicklung neuer Software gehören. Software-Prozessmodelle sind die abstrakten Repräsentationen dieser Prozesse.
2. Prozessmodelle beschreiben die Organisation von Softwareprozessen. (Wasserfall, inkrementelle Entwicklung, Wiederverwertungs-orientierte Entwicklung)
3. Requirements Engineering: Entwicklung einer Software-Spezifikation

4. Design und Implementierung wandeln die Anforderungs-Spezifikation in ein ausführebares Softwaresystem um.
5. Validierung: Erfüllung der Sytem-Spezifikation und der Kundenwünsche wird überprüft
6. Evolution: Veränderung existierender Systeme zur Erfüllung

3.4.6 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

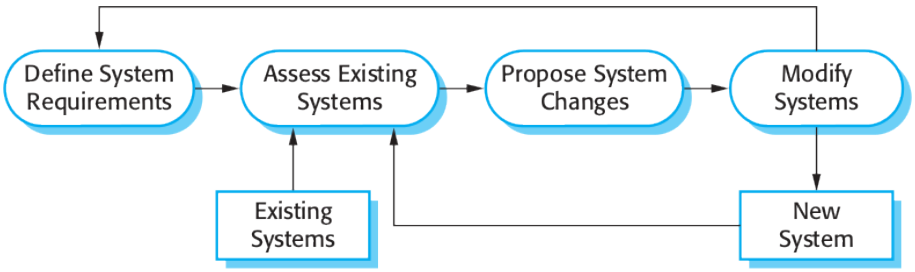

Abbildung 3.9: Software-Evolution

3.5 Bewältigung von Veränderungen

Veränderungen führen zu Nacharbeit. Die Veränderungskosten beinhalten Nacharbeit und Implementierung neuer Funktionalitäten

3.5.1 Kostenreduzierung

- **Change avoidance:** vorwegnehmen möglicher Veränderungen
- **Change tolerance:** Gesamtprozess kann Veränderungen leicht umsetzen

3.5.2 Prototyping

- Demonstration von Konzepten und ausprobieren von Designmöglichkeiten
- 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 untersuchen und den Entwurf der Benutzeroberfläche zu unterstützen

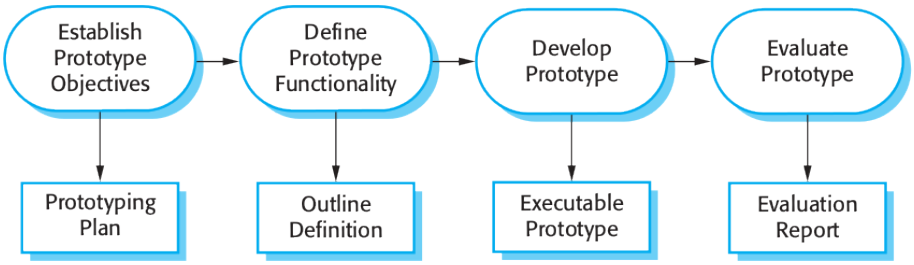

Abbildung 3.10: Prototyping

3.5.3 Spiralmodell nach Boehm:

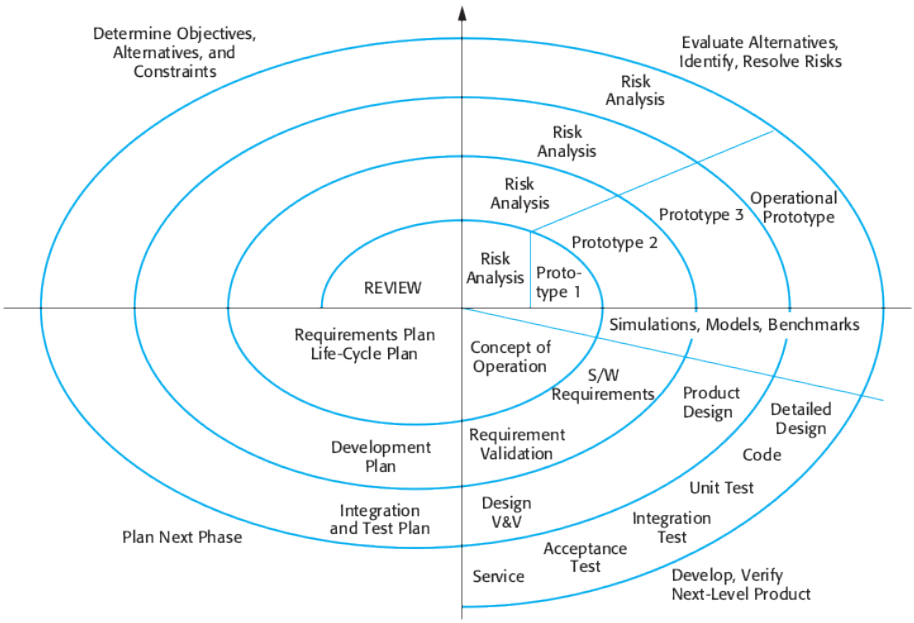

Abbildung 3.11: Spiralmodell nach Boehm

- 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
- **Risiken** werden während dem Prozess angesprochen und geklärt

- Nutzung:
  - sehr einflussreich um Menschen zu helfen über Iterationen in Softwareprozessen und die Eifnührung des risikoorientierten Ansatzes für die Entwicklung nach zu denken
  - Praxis
    → Modell wird selten verwendet
    → wird zur Veröffentlichung der praktischen Softwareentwicklung verwendet

3.5.4 Rational Unified Process (RUP):

- Generisches Verfahren
- abgeleitet von der Arbeit an der UML
- Vereinigt Elemente aus allen allgemeinen/generischen **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:**
    zeigt Entwicklungsphasen über die Zeit
  - **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

3.5.5 Key Points

- Prozesse sollten Verfahren für den Umgang mit Veränderungen einbeziehen
- Iterative Entwicklung und Auslieferung: Veränderungen können umgesetzt werden, ohne das Gesamtsystem zu beeinträchtigen
- Boehms Spiralmodell nimmt Rücksicht auf Risiken und Neuplanungen/Überarbeitungen
- RUP ist ein modernes allgemines Vorgehensmodell, das zwar in Phasen unterteil ist **(Konzeption, Entwurf, Konstruktion und ÜBergabe)**, jedoch die Aktivitäten (Anforderungsanalyse, Analyse und Entwurf) von den Phasen trennt

4 Projekt Management

4.1 Einleitung

4.1.1 Erfolgskriterien

1. Ausliefern der Sfotware zur vereinbarten Zeit
2. Budget einhalten
3. Software erfüllt Anforderungen
4. gute Personalführung

- Besonderheiten:
  - Software ist nicht greifbar, Fortschritt kann nicht direkt beobachtet werden
  - Softwareprojekte unterscheiden sich stark voneinander → Probleme schwer abschätzbar
  - **Softwareprozesse sind variabel un organisationssepzifisch**, Probleme sind immer noch schwer vorherzusagen

4.1.2 Management activities

1. **Angebot schreiben**
   - Beschreibung der Projektziele, Vorgehenweise . . . → einholen von Aufträgen
2. **Projektplanung**
   - Planung, Abschätzung des Aufwandes und Erstellung eines Zeitplanes
3. **Berichterstattung**
   - Kunden und Management müsse über Fortschritt des Projektes unterrichtet werden
4. **Risikomanagement**
   - abschätzen, beobachten und behandeln von Risiken
5. **Personalführung**
   - Auswahl von Mitarbeitern und Herstellung von produktiven Arbeitspozessen

4.2 Risikomanagement

- Identifikation von Risiken und entwickeln von Plänen zur Minimierung ihrer Effekte auf das Projekt
- **Risiko: Warscheinlichkeit, dass negative Umstände eintreten**
- Risikobereiche
  - **Projektrisiken** beeinflussen Zeitplan oder Mittel
  - **Produktrisiken** beeinflussen Qualität oder Leistung der entwickelten Software

  - **Business risks** beeinflussen das Unternehmen, welches die Software entwickelt

- Vier Schritte:
  1. **Risikoerkennung**
     (Projekt-, Produkt- und Unternehmensrisiken)
  2. **Risikoanalys**
     (Wahrscheinlihckeit und Folgen)
  3. **Risiko-Planung**
     (Vermeiden oder klein halten)
  4. **Risikobeobachtung**
     (Beobachtung während dem Projekt)

4.2.1 Key Points:

- Gutes Projektmanagement ist entscheidend für die Einhaltung von Zeitplan und Budget
- Software ist nicht greifbar, Projekte können neuartig sein (keine Erfahrungswerte), Sfotwareentwicklung ist eine vergleichsweise neue Disziplin
- Risikomanagement beinhaltet Identifikation und Beurteilung von Projektrisiken ur Abschätzung der Wahrscheinlichkeit ihres Aufretens und ihrer möglichen Auswirkungen auf das Projekt
- Vermeidung, Kontrolle oder umgang von wahrscheinlichen Risiken muss in die Planung miteinbezogen werden

4.3 Personalführung

4.3.1 Faktoren

- **Konsequenz:** Temamitglieder sollten alle gleich behandelt werden
- **Respekt:** Unterschiedlihe Fähigkeiten der Teammitglieder sollten berückstichtigt werden
- **Einbeziheung:** Alle Teammitglieder sollten in den Prozess miteinbezogen werden
- **Ehrlichkeit:** Ehrlichkeit bei Lob & Kritik

4.3.2 Motivation

- Organisation von Arbeit und ARbeitsumgebung, so dass Mitarbeiter zu effektiver ARbeit motiviert werden
- Verschiedene formen von Motivation, je nach zugrundliegendem Bedürfnis
  - Grundbedürfnisse (Essen, Schlafen, ...)
  - persönliche Bedürfnisse (Respekt, Selbstbewusstsein, ...)
  - soziale Bedürfnisse (akzeptiert werden, ...)
- Motivation sollte die verschiedenen Persönlichkeitstypen berücksichtigen

4.3.3 Gruppenzusammensetzung

- Alle Persönlichkeitstypen sollten gleichermaßen vorhanen sein
  - Aufgaben-orientiert: die Motivation für die Tätigkeit ist die Tätigkeit selbst
  - Selbstbezogen: Arbeit ist Mittel zum Zweck
  - Interaktions-orientiert: Hauptmotivation ist Anwesenheit und Mitarbeit von Kollegen
- Interaktionsorientierte Persönlichkeiten besonders wichtig

4.3.4 Key Points:

- Menschen werden durch utnerschiedliche Dinge motiviert (Interaktion, Anerkennung, Möglichkeiten zur persönlichen Weiterentwicklung)
- Gruppen sollten relativ klein und zusammenhängend sein, Gruppenzusammenhalt abhängig von Mitgliedern, Organisation und Kommunikation
- Gruppeninterne Kommunikation abhängig vom Status der Gruppenmitglieder, Gruppengröße, Geschlechterverteilung, Persönlichkeiten und Kommunikationswegen

4.4 Projektplanung

- Aufgabe einteilen und diese den Teammitgliedern zuweisen, Probleme vorhersehen und Lösungsmöglichkeiten vorbereiten
- Projektplan als Kommunikationsmittel mit Teammitgliedern und Kunden, bewerten des Fortschritts
- **Planungsstufen:**
  - Angebotsstufe (nur Entwurf der Software Requirements → Preisgestaltung)
  - Anfangsstufe des Projektes
  - regelmäßig während des Projektes

4.4.1 Preisgestaltung: Faktoren

- Kostenabschätzung für den Entwickler
- organisatorische, wirtschaftliche, politische und unternehmensbezogene Erwägungen

1. Marktchancen (niedrigerer Preis → Eroberung neuer Marktsegmente)
2. Unsicherheit bei der Kostenabschätzung (Preis zur Sicherheit höher angesetzt)
3. Vertragsbedingungen (niedrigerer Preis, wenn Rechte am Quelcode beim Entwickler bleiben)
4. Unbeständigkeit der Anforderungen (zunächst niedriger Oreis, nach Fertigstellung hohe Gebhren für die Umsetzung von Änderungen)
5. finanzielle Lage (besser geringer oder kein Gewinn als Insolvenz)

4.4.2 Plangesteuerte Entwicklung

- Alle Prozessaktivitäten von Anfang an geplant, Fortschritt an dieser Planung gemessen
- **Vorteile**
  - verbesserte Organisation durch frühe Planung
  - mögliche Probleme und Abhängigkeiten fallen vor Projektbeginn auf
- **Nachteile**
  - viele frühe Entscheidungen müssen später revidiert werden (veränderte Anforderunger/bedingungen)
- **Projektplan**
  - **Was** wird **von wem wann mit welchem Werkzeugen** getan?

Abschnitte:

1. Einleitung
2. Projektorganisation
3. Risikoanalyse
4. Hardware- und Softwareanforderungen
5. Arbeitseinteilung
6. Zeitplan
7. Überwachungs- und Berichterstattungsmechanismen

Zeitplanung

- Abschätung der von jedem Arbeitsschritt benötigten Zeit und desbenötigten Arbeitsaufwandes.
- Abschätzung der benötigten Ressourcen (Hardware, finanziell)

Arbeitsschritte:

1. Aufteilung des Projektes in Arbeitsschritte, Abschätzung der benötigten Zeit und Ressourcen
2. Arbeitsschrite gleichzeitig ausführen lassen → optimale Ausnutzung der Arbeitskraft
3. Minimierung von Abhängigkeiten zwischen den Arbeitsschritten → weniger Zeitverlust durch Wartezeiten

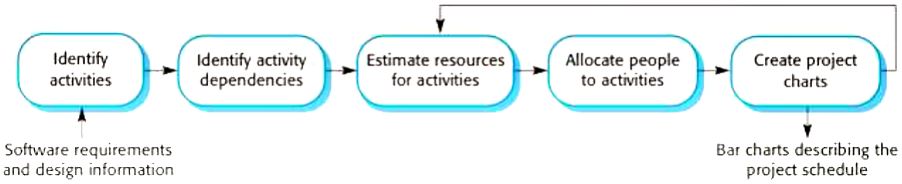

Abbildung 4.1: Project schedulding

Milestones und Deliverables:

- **Milestones:** Punkte im Zeitplan, dienen zur Bewertung des Fortschrittes
- **Deliverables:** Erzeugnisse, die an den Kunden ausgeliefert werden (Requirements documents)

Probleme:

- Abschätzung des Ausmaßes von Problemen und damit der Kosten für ihre Lösung ist schwierig
- Produktivität ist nicht proportional zur Auswahl der Mitarbeiter
- Einsatz von zusätzlichen Arbeitskräften bei einem verspäteten Projekt führt zu zusätzlicher Verspätung (Kommunkations-Overhead)
- Murphy's Law: möglichst alle Eventualitäten einplanen

4.4.3 Agile Planung: Schritte

- **Agile Planung:** inkrementelle Planung, Prozess kann bei veränderten Anforderungen abgeändert werden
- **Planung der Veröffentlichung:** mehrere Monate im Voraus, bestimmt die in der Veröffentlichung inbegriffenen Funktionen
- **Planung der Iterationen:** kurzfristig, Planung es nächsten Inkrements (2-4 Wochen Arbeitszeit)
- **Extreme Programming (XP):** Story-basierte Planung

4.4.4 Key Points:

- Preis für Software wird nicht nur durch Entwicklungskosten bestimmt, sondern kann durch Markt- oder Unternehmensprioritäten beeinflusst werden.
- Planungsgesteuertes entwickeln wird um inen kompletten Projektplan organisiert, definiert Projektaktivitäten, Aufwand, Zeitplan und Aufgabenverteilung
- Erstellung des Zeitplans mit Hilfe von graphischen Darstellungen des Projektplans
- XP-Planspiel beteiligt das gesamte Team an der Projektplanung. Plan wird inkrementell entwickelt, wird bei auftretenden Problemen angepasst (Reduzierung von Funktionalitäten anstatt Verzögerung der Auslieferung)

4.5 Techniken zur Aufwandsabschätzung

Zwei Arten:

- *Erfahrungsbasiert:* Abschätzung des benötigten Aufwandes abhängig von Erfahrung des Projektleiters und des Anwendungsbereiches der Software
- *Algorithmische Kostenmodellierung:* formelähnliche Herangehensweise zur berechnung des Arbeitsaufwandes, basierend auf Abschätzung von Projektattributen (Größe, Erfahrung der Mitarbeiter, ...)

4.5.1 Algorithmische Kostenmodellierung

- Mathematische Funktion aus Produkt, Projekt, Prozessatributen. Werte werden vom Projektleiter abgeschätzt
- $A \times Size^B \times M$
  - A: organisationsabhägige Konstante
  - B: spiegelt den uvnerhätlnismäßigen Aufwand für große Projekte wieder
  - M: Multiplikator, spiegelt Produkt-, Pozess- und Personalattribute wieder
- Häufigstes Produktattribut: Codegröße
- Modelle meist ähnlich, aber unterschiedliche Werte für A, B und M

4.5.2 Genauigkeit der Abschätzung

- Größe einer Sfotware erst nach Fertigstellung bekannt
- beeinflussende Faktoren:
  - Verwendung von COTS und bereits existierenden Komponenten
  - Programmiersprache
  - Verteilung des Systems
- je weiter der Entwicklungsprozess fortgeschritten ist, desot genauer kann die codegröße abgeschätzt werden
- Abschätzungen der zu B und M gehörenden Faktoren subjektiv

COCOMO 2

- empirisches Modell, erfahrungsbasiert
- gut dokumentiert, unabhängig
- mehrfah überarbeitet
- berückstichtigt verschiedene Herangehensweisen an softwareentwicklung, Wiederverwendung etc.
- enthält eine Anzahl von Untermodellen → zunehmend detailliertere Abschätzungen
- Untermodelle:
  1. **Application-composition-Modell**
     zusammensetzen von Software aus bereits existierenden Teilen
  2. **Frühes Designmodell**
     Anforderungen verfügbar, aber Designprozess noch nicht begonnen
  3. **Reuse-Modell**
     Berechnung des Aufwandes, wiederverwendbare Komponenten zu integrieren
  4. **Post-architecture-Modell**
     nach Design der Systemarchitektur → mehr Infromation über das System verfügbar

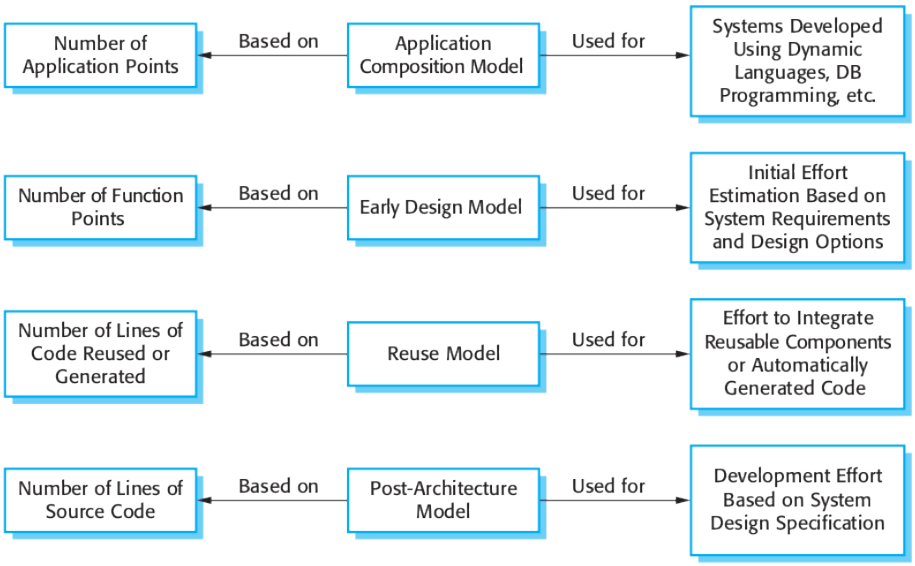

Abbildung 4.2: COCOMO estimation models

Application composition-Modell

- Prototypenentwicklung, Projekte mit viel Wiederverwendung
- **standard estimates of developer productivity in application (oject) points/month**
- Nutzung von CASE berücksichtigen
- **Gleichung:** $PM = (NAP \times (1 - \%reuse/100))/PROD$
  - PM: Aufwand (Monate Arbeitszeit pro Person)
  - NAP: Anzahl der application points
  - $PROD$: Produktivität

Frühes Designmodell

- Abschätzung nach Einigung über Anforderungen
- basiert auf Standartformel für algorithmische Modelle:
  - $PM = A \times Size^B \times M$
    - $M = PERS \times RCPX \times RUSE \times PDIF \times PREX \times FCIL \times SCED$
    - A: beginnt bei 2.94 [KLOC]
    - B: 1.1 - 1.24 (abhängig von Neuartigkeit des Projektes, Flexibilität der Entwicklung Risikomanagement, Reife des Prozesses)

Multplikatoren

- Produktattribute
  - geforderte Eigenschaften der Software
- Computer-Attribute
  - Eisnchränkungen durch Hardware

- Personalattribute
  - Erfahrung und Fähigkeit der Entwickler
- Projektattribute
  - Eigenschaften des Entwicklungsprozesses

4.5.3 Key Points:

1. Abschätzungs-Techniken für Software sind erfahrungsbasiert (Projektleiter bewerten den benötigten Aufwand) oder algorithmisch (Aufwand berechnet aus anderen abgeschätzten Projektparamtern)
2. COCOMO 2 - Kostenberechnungsmodell: algorithmisches Modell, das Projekt-, Produkt-, Hardware- und Personalattribute, sowie Produktgröße und -komplexität beachtet

5 Qulalitäts-Management

5.1 Software-Qualitätsmangement

Hauptanliegen:

1. organisatorische Ebene: schaffen eines Gerüstes aus organisatorischen Prozessen und Standards
2. Projekt-Ebene: Anwendung spezigfischer QM-Prozesse, Kontrolle von deren Einhaltung
3. Projekt-Ebene: Erstellung eines Qulitätsplanes, legt Ziele, Prozesse und Standards fest

Qulitätsmangement

- QM-Team muss unabhängig vom Entwicklerteam sein
- Kontrolle von Devlierables zur Sicherstellung der Unternehmensstandards

5.1.1 Qualitätsplan

- Produktqualität, Qualitätsbewertung, Qualitätsmerkmale, Standards
- Bewertungsprozess
- Auswahl oder Neudefinition von Unternehmensstandards
- Aufbau:
  - Produkteinführung
  - Produktplan: Veröffentlichungsdaten, Verantwortlichkeiten, ...
  - Qualitätsziele
  - Risiken und Risikomanagement

Probleme:

1. Spannung zwischen den Anforderungen von Kunde und Entwickler an die Qualität
2. eindeutige Formulierung von Qualitätsanforderungen schwer

5.1.2 Software-Qualitätsattribute

Sicherheit	Verständlichkeit	Protabilität
Schutz	Testbarkeit	Benutzbarkeit
Zuverlässigkeit	Anpassbarkeit	Wiederverwenbarkeit
Ausfallsicherheit	Modularität	Effizienz
Robustheit	Komplexität	Elernbarkeit

Prozess- und Produktqualität

- **Produktstandards** definieren Charakteristiken, die alle Software-Komponenten aufweisen sollen
- **Prozessstandards** definiere, wie der Software-Prozess ausgeführt werden soll

Produkt-Standard	Prozess-Standards
Form des Design-Reviews	Ausführung des Desing-Reviews
Struktur des Anforderungsdokumentes	Einreichen von neuem Code
Format der Methodenköpfe	Versions-Veröffentlichungsverfahren
Programmierstil	Annahme des Projektsplanes
Format des Projektplanes	Änderungskontrolle

5.1.3 ISO 9001 - Struktur

- Internationaler Satz von Standards → Grundlage für die Entwciklung von Qualitätsmanagement-Systemen
- gelten für Unternehmen, die Produkte entwerfen, entwickeln und unterhalten
- ISO 9001 speziell für Softwareentwicklung
  - allgemeine Qualitätsprinzipien
  - allgemine Qualitätsverfahren
  - Auflistung von Unternehmensstandards und -verfahren

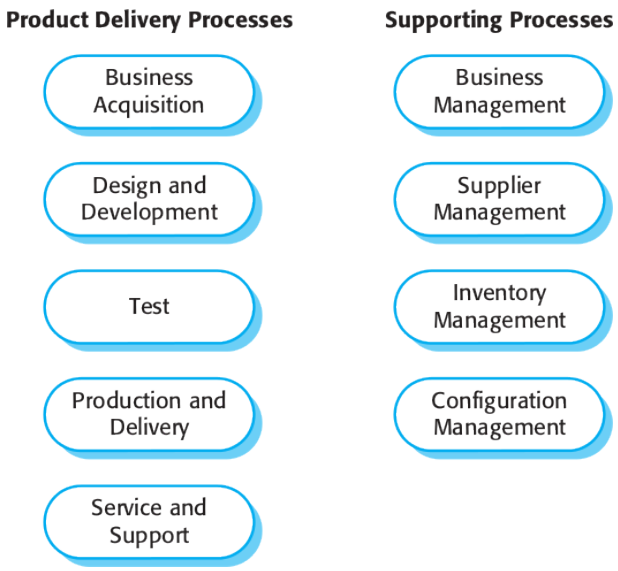

Abbildung 5.1: ISO 900 Kern-Prozesse

- ISO 9001-Zertifikation
  - Qualitätsmanagement-Handbuch des Unternehmens
  - Externe Stelle bescheinigt die Einhaltung der ISO 9001-Standards
  - zunehmende flexiblie Handhabung notwendig

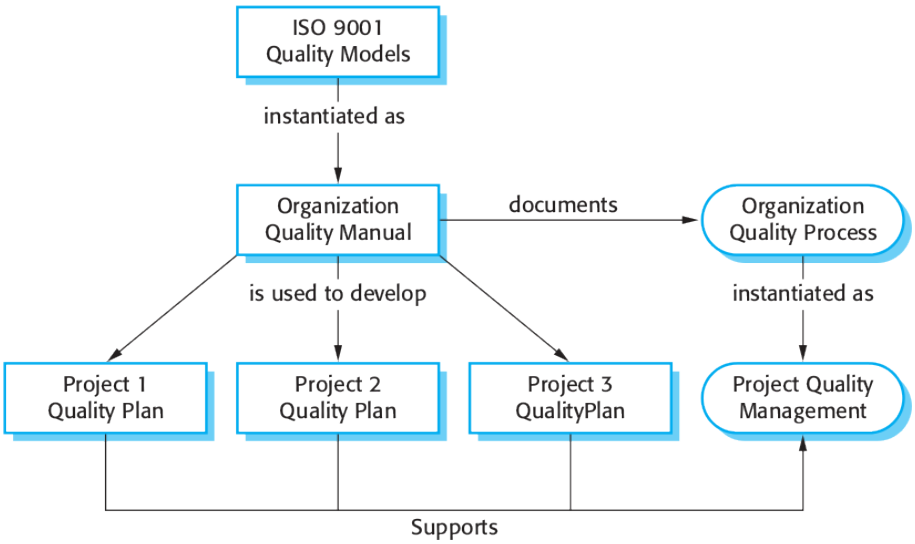

Abbildung 5.2: ISO 9001-Qualitätsmanagement

5.2 Reviews and Inspections

- Gruppe kontrolliert Prozess oder System und Dokumentation → mögliche Probleme finden
- Akzeptanz eines Iteratinsschrittes durch die Projektleitung dokumentiert
- geprüft werden können: Code, Design, Spezifikationen, Testpläne, Standards, ...
- **Sollten durch Checklisten gesteuert werden!**

5.2.1 Reviews agiler Methoden

- Prüfprozess informell
- Extreme Programming: Pair Programming stellt kosntante Kontrolle und Überprüfung des codes durch das andere Teammitglied sicher.

5.2.2 Software measurement and metrics

- Software measurement: Entwicklung eines numerischen Wertes für Eigenschaften eines Softwareproduktes ode -prozesses
- Ermöglicht objektiven Vergleich zwischen Techniken und Prozessen

Software metric

- jede Form von Vermessung eines Software-Systems, -prozesses oder -dokumentation
  - Lines of code, Fog Index, Antahl der Arbeitstage pro Person, ...
- Software kann in Zahlen ausgedrückt werden
  - Vorhersage von Produkteigenschaften, Kontrolle der Softwareprozesse
  - Identifikation regelwiedriger Komponenten

Dynamische und statische Product Metrics

- **Dynamische Metriken:** während der Ausführung aufgenommen
  - verwandt mit Qualitätseigenschaften → Effizienz, Zuverlässikgeit
  - einfach gemessen werden können: response time (Performance-Eigenschaft), Fehlerfrequenz (Zuverlässigkeits-Eigenschaft)
- **Statische Metriken:** berechnet aus Sstemdarstellungen
  - indirekte Verwandschaft mit Qualitätseigenschaften
  - Beurteilung von Komplexität, Verständlichkeit und Unterhaltbarkeit

5.3 Prozessverbesserung

Steigerung der Produktqualität und/oder Kostenreduktion, Verkürzung der Entwicklungszeit

Herangehensweisen:

- **Prozessreife-Ansatz:** Prozess- und Projektmangement-Verbesserung → *good practice*
  - Level der Prozessreife: Ausmaß der Anwendung von guten technischen und Führungsmethoden
- **Agiler Ansatz:** iterative Entwicklung und REduktion von Overheads im Softwareprozess
  - schnelle Auslieferung und REaktion auf Anforderungsänderungen

5.3.1 Prozessqualität

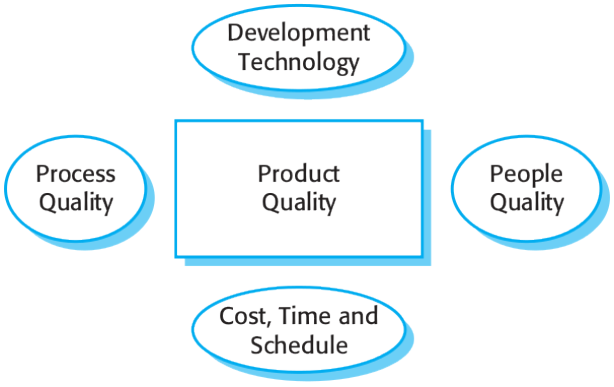

Abbildung 5.3: Einflüsse auf die Softwarequalität

- Großes Projekt, mittlere Fähigkeiten: Entwicklungsprozess bestimmt die Produktqualität
- Kleines Projekt: Fähigkeiten und Entwicklungsmethoden bedeutend
- realistischer Zeitplan essentiell

Beispiele für Prozess-Messgrößen:

- Dauer von Prozessaktivitäten
- Benötigte Ressourcen
- Häufigkeit des Auftretens bestimmter Ereignisse

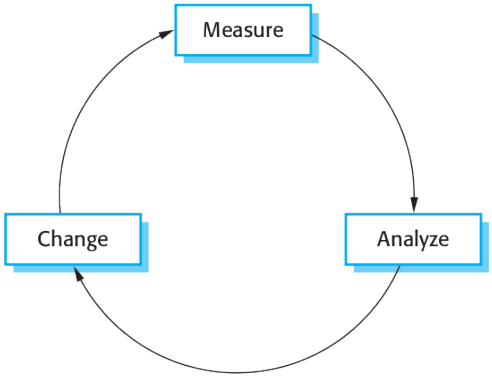

Abbildung 5.4: Prozess improvement cycle

Prozessanalyse-Techniken:

- Veröffentlichte Prozessmodelle und -standards
  - existierenden Modelle können nach Bedarf erweitert und verändert werden
- Fragebögen und Interviews
  - Gefahr: nicht wahrheitsgemäße Antworten
- Ethnographische Analysen (Informationsgewinnung durch Beobachtung)

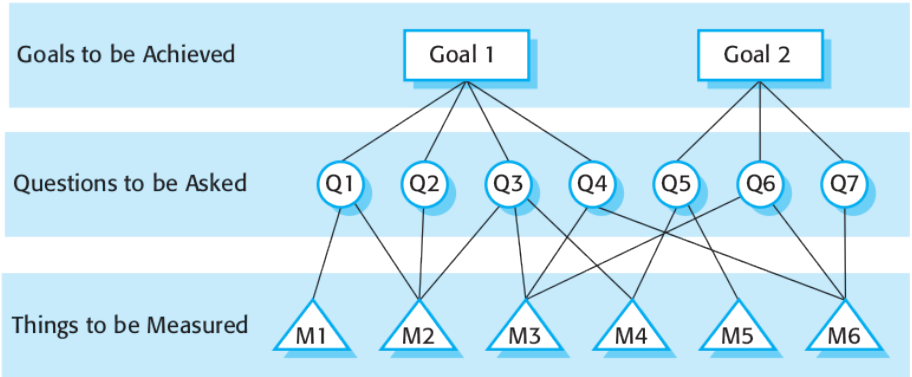

Abbildung 5.5: CQM paradigm (Goals, Questions Measurements)

- Modifikation existierender Prozesse
- beinhaltet:

  - Einführung neuer Praktiken, Methoden, Prozesse
  - Änderungen der Anordnung von Prozessaktivitäten
  - Einführung oder Entferung von Deliverables
  - Einführung neuer Rollen und Verantwortlichkeiten
  - ...

- Probleme:
  - Wiederstand gegen Veränderungen
  - beständigkeit von Veränderung (Institutsionalisierung von Veränderungen sicher Beibehaltung)

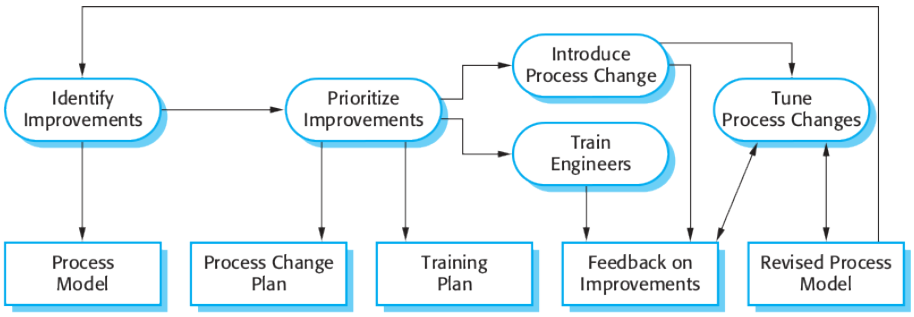

Abbildung 5.6: Prozessänderung

Abbildungsverzeichnis