Software Engineering

Felix Scholzen

15.07.2024

Inhaltsverzeichnis

1 Einführung

1.1 Ziele des Software Engineering

1. **Qualität**:
   - Zuverlässigkeit
   - Wartbarkeit
   - Benutzerfreundlichkeit
   - Performanz
   - Effizienz
2. **Kosten**:
   - Entwicklungskosten
   - Wartungskosten
   - Vertriebskosten
3. **Zeit**:
   - Verkürzung der Entwicklungszeit durch mehr Personal
   - Spezialisten und motivierte Teams
4. **Wirtschaftlichkeit**:
   - Umsatz und Kosteneinsparung durch effiziente Softwarelösungen

2 Ethik

2.1 Internationale Organisationen

- ACM(Association for Computing Machinery)
- IEEE (Institute of Electrical and Electronic Engineers)
- British Computer Society

"Software Engineering Code of Ethics and Professional Practice"
`https://ethics.acm.org/code-of-ethics/software-engineering-code/`

2.2 8 Prinzipien des ACM/IEEE Code of Ethics

1. **Public**:
   - Handeln im öffentlichen Interessen
   - Volle Verantwortung für seinen Code übernehmen
   - Genehmigen Sie Software nur, wenn sie sicher ist, die Spezifikationen erfüllt, geeignete Tests besteht und die Lebensqualität nicht beeinträchtigt, die Privatsphäre nicht verletzt und die Umwelt nicht schädigt
2. **Client and Employer**:
   - Interessen von Kunden und Arbeitgebern wahren
3. **Product**:
   - Höchste professionelle Standards für Produkte sicherstellen
4. **Judgement**:
   - Integrität und Unabhängigkeit im beruflichen Urteil bewahren
5. **Management**:
   - Ethisches Management von Softwareentwicklung und -wartung
6. **Profession**:
   - Integrität und Reputation des Berufsstandes fördern
7. **Colleagues**:
   - Fair und unterstützend gegenüber Kollegen sein
8. **Self**:
   - Lebenslanges Lernen und ethische Praxis fördern

2.3 Ethikfragen im Software Engineering

- Ist die Verbreitung von Viren/Trojanern ein ethisches Problem?
- Ist fehlerhafter Code ein ethisches Problem?
- Ist schwer lesbarer Code ein ethisches Problem?
- Ist das Unterschätzen der Schwierigkeit eines SW-Projekts ein ethisches Problem?
- Ist das Unterschätzen der Kosten eines Softwareprojekts ein ethisches Problem?
- Ist es ein ethisches Problem, nicht immer die neuesten Technologien zu nutzen?

3 Projektphasen

3.1 Wie kommt man zu einem Projekt?

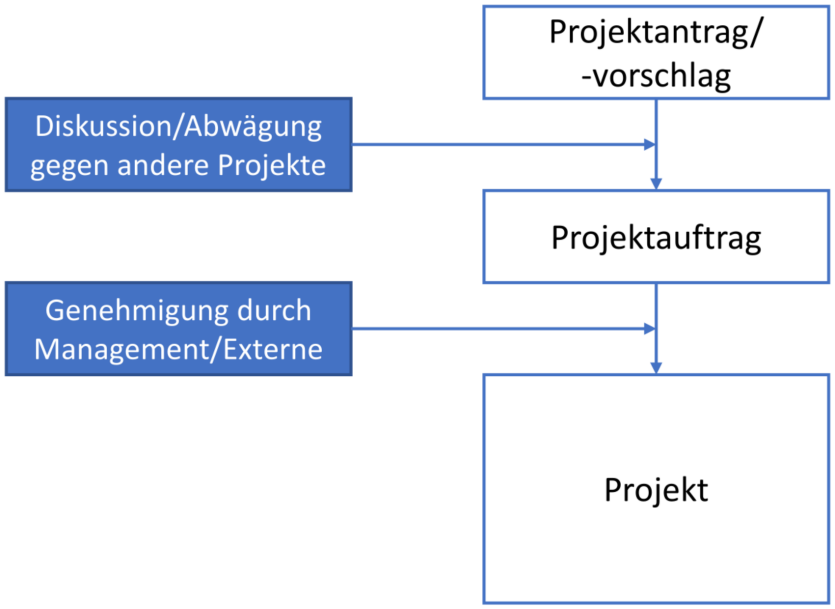


Abbildung 1: Projektantrag Ablauf

3.1.1 Projektantrag

- **Ziel:** Darlegung der Projektidee
- **Inhalte:**
  - Ausgangslage (Status quo, welches Problem wollen wir lösen?)
  - Ziele
  - Kosten/Nutzen
  - Organisation
- **Ausprägungen:**
  - Interne Projekte
  - öffentlich geförderte Projekte

3.1.2 Projektauftrag

Offizieller Startschuss für das Projekt. Projektauftrag ist das erste Projektdokument.
!!!Kein Projekt ohne Projektauftrag!!!

- **Inhalte:**
  - Projektbezeichnung
  - Auftraggeber
  - Beginn und Ende
  - Kurzbeschreibung
  - Unternehmensbedarf
  - Ziele
  - Projektergebnisse
  - Budget
  - Projektleiter
  - Team
  - Annahmen
  - Beschränkungen
  - Terminvorgaben

Unternehmen haben sehr unterschiedliche Handhabung im Umgang mit Projektantrag und Projektauftrag. Er hängt von Organisationsstruktur und Kultur des Unternehmens ab. Bei mittelgroßen/großen Unternehmen gibt es normalerweise Vorlagen für den Projektauftrag!

3.2 Phasen eines Softwareprojekts

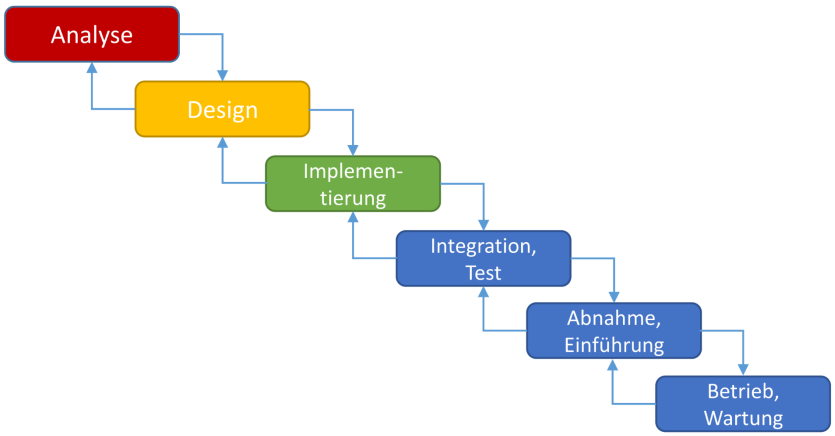


Abbildung 2: Projektphasen Ablauf

Analyse

- Aufstellen von Leistungen, Einschränkungen und Zielen des Systems
- In Zusammenarbeit mit den Systembenutzern/dem Kunden
- Anschließend detailliertere Beschreibung und Definition
- Ergebnis dient als Systemspezifikation

Design (Entwurf)

- Anforderungen werden in Hard- oder Softwaresysteme aufgeteilt
- Dazu wird übergeordnete Systemarchitektur festgelegt
- Beim Software-Design geht es um Erkennen und Beschreiben:
  - Der grundlegenden (abstrakten) Softwaresysteme und
  - Deren Beziehung zueinander

Implementierung (und Modultests)

- Umsetzung des Softwareentwurfs durch Programme oder Programmeinheiten
- Testen der Module auf Einhaltung Ihrer Spezifikation

Integration, Test

- Zusammenführung/Integration einzelner Programme oder Programmeinheiten
- Test des "Ganzen" auf Einhaltung der Anforderungen
- Auslieferung an Kunden

Abnahme, Einführung

- Abnahmetests in mehreren Stufen
  - Technische Tests
  - Last-/Performancetests
  - Fachliche Tests
  - User-Acceptance Tests, ...
- Auslieferung an den Kunden
- Ggf. Anwenderschulungen, Informationsmaßnahmen bis hin zu Marketing für die neue Software

Betrieb, Wartung

- Längste Phase im Software-Lifecycle
- System wird installiert und zur Nutzung freigegeben
- Elemente der Wartung:
  - Beheben von Fehlern
  - Verbesserung der Implementierung von Systemeinheiten
  - Verbesserung des Systems, falls neue Anforderungen entstehen
- Teil der Wartung häufig noch in Projektkosten enthalten
- Später Wartungsverträge

3.3 Allgemeine Hinweise zu den Projektphasen

- Phasen nicht notwendigerweise sequentiell
- Tätigkeiten innerhalb der Phasen erfordern verschiedene Qualifikationen
- ABER: nicht immer ist (vollständige) Durchführung aller Phase sinnvoll
- Tätigkeiten müssen koordiniert werden → Projektmanagement
- Softwareprojekte laufen immer in Teams ab (sowohl innerhalb des Unternehmens als auch mit Kooperationspartnern/Kunden)

→ **Kommunikation innerhalb des Teams und zwischen Team und Kunden ist extrem wichtig**

3.4 Dokumente (Artefakte) pro Phase

Jede Phase bringt Dokumente (Artefakte) hervor, die Input für die folgende Phase sind

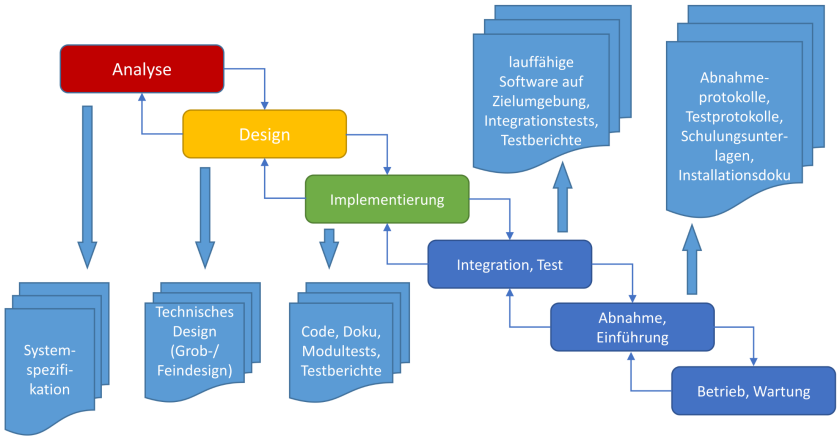


Abbildung 3: Dokumente der einzelnen Phasen

4 Requirements Engineering (Anforderungsanalyse)

4.1 Überblick und Bedeutung

Definition

- Requirements Engineering ist der Prozess des Herausfindens, Analysierens, Dokumentierens und Überprüfens der Anforderungen an ein System
- **Ziel:** Sicherstellen, dass alle relevanten Anforderungen bekannt und verstanden sind, und dass Stakeholder diesen zustimmen

Bedeutung der Analysephase

- Fehler in der Analysephase führen zu hohen Kosten und Verzögerungen
- 60% der Softwareprojekte scheitern aufgrund von Analysefehlern
- 50% der Ausfälle im industriellen Sektor sind auf Software-Fehler zurückzuführen

Grundsatz: *Frühzeitig Fehlererkennung/-behebung spart Zeit, Kosten, Ärger*

4.2 Hauptaufgaben

Anforderungen ermitteln

- Systemkontext festlegen
- Anforderungen von Stakeholdern sammeln

Anforderungen dokumentieren

- Beschreibung der Anforderungen in Lasten- und Pflichtenheft

Anforderungen prüfen und abstimmen

- Überprüfung der Anforderungen auf Vollständigkeit, Konsistenz und Eindeutigkeit
- Stakeholder-Review und formale Abnahme

Anforderungen verwalten

- Änderungen nachverfolgen
- Versionierung der Anforderungen

4.3 Requirement Dokumente nach IEEE 830-1998

1. Einleitung

(a) Zielsetzung (Vision)
(b) Produktziele
(c) Definitionen, Akronyme, Abkürzungen
(d) Referenzen
(e) Überblick

2. Übersichtsbeschreibung

(a) Produkt-Umgebung
(b) Produkt-Funktionen
(c) Benutzer-Eigenschaften
(d) Restriktionen
(e) Annahmen und Abhängigkeiten

3. Spezifische Anforderungen

(a) Externe Schnittstellenanforderungen
(b) Funktionale Anforderungen
(c) Leistungsanforderungen
(d) Entwurfsrestriktionen
(e) Eigenschaften des Softwaresystems
(f) Andere Anforderungen

4.4 Visionen, Ziele und Rahmenbedingungen

Bevor die Anforderungen und Rahmenbedingungen aufgenommen werden, sollten Visionen und Ziele formuliert werden.

4.4.1 Visionen

- beschreibt, **was** erreicht werden soll, aber **nicht wie**
- hat **geringe Detailtiefe**; dient als **Leitgedanke**
- wird im Rahmen eines Projektauftrags formuliert, der genehmigt werden muss

Beispiele:

Vision für ein Türsteuergerät für einen Fensterheber:
/V20/ *Die Fensterheberkomponente soll das komfortable Heben und Senken der Seitenfenster eines Fahrzeugs ermöglichen.*
Vision für eine Kundendatenbank:
/V20/ *Die Kundendatenbank soll als single point of truth alle Stammdaten der Kunden enthalten.*

4.4.2 Ziele

Ausgehend von einer Vision dienen Ziele dazu, die **Vision** zu **verfeinern** und zu **operationalisieren**

Beispiel: Ziel für den Fensterheber

Vision für ein Türsteuergerät für einen Fensterheber:
/Z20/ *Die erwarteten Stückzahlen betragen 20.000 Einheiten p.a.*

Regeln zur Zielformulierung:

1. Kurz und prägnant formulieren
2. Aktivformulierung
3. Überprüfbare und realistische Ziele formulieren
4. Nicht überprüfbare Ziele verfeinern
5. Mehrwert eines Ziels hervorheben
6. Begründung für das Ziel liefern
7. Keine Lösungsansätze formulieren

Eigenschaften von Zielen:

- vollständig
- korrekt
- konsistent gegenüber anderen Zielen und in sich konsistent
- testbar
- verständlich für alle Stakeholder
- umsetzbar/realisierbar
- notwendig
- eindeutig und positiv formuliert

4.4.3 Rahmenbedingungen

Legt organisatorische und technische Restriktionen für das Softwaresystem bzw. den Entwicklungsprozess fest.

- **Organisatorische Rahmenbedingungen:**
  - Anwendungsbereiche (z.B. Textverarbeitung im Büro)
  - Zielgruppen (z.B. Sekretärinnen, Schreibkräfte)
  - Betriebsbedingungen (z.B. Büroumgebung, mobiler Einsatz)
- **Technische Rahmenbedingungen:**
  - Technische Produktumgebung
  - Anforderungen an die Entwicklungsumgebung

4.5 Anforderungen

Änforderungen legen fest, was man von einem Softwaresystem als Eigenschaften erwartet"
- *Helmut Balzert*

4.5.1 Funktionale Anforderungen

- Beschreiben, was ein System tun soll (Dienste, Reaktionen auf Eingaben, Verhalten)

4.5.2 Nicht-funktionale Anforderungen

- Beschränkungen
- Beziehen sich eher auf das Gesamtsystem als auf einzelne Dienste

Beispiele: Nicht-funktionale Anforderungen

Technische Anforderungen

- Portierbarkeit
- Schnittstellen

Qualitäts Anforderungen

- Wartbarkeit
- Effizienz
- Zuverlässigkeit
- Änderbarkeit

Unternehmensanforderungen

- Lieferanforderungen
- Vorgehensanforderungen

Ethische Anforderungen

- *Siehe Kapitel 2.*

Rechtlich-vertragliche Anforderungen

- Sicherheit
- Datenschutz
- Gesetze

4.6 Stakeholder

→ Jemand der Einfluss auf die Anforderungen hat, da er vom System betroffen ist.
SSystembetroffener"

- Sind Quellen für Anforderungen
- Übersehen eines Stakeholders führt zu lückenhaften Anforderungen

Zum Biespiel: Kunden, Mitarbeiter, Zulieferer, Eigentümer, ...

4.6.1 Stakeholderliste

Name/Rolle	Beschreibung	Konkreter Vertreter	Wissensgebiete	Verfügbarkeit	Begründung
Kneipen-Management	Nennt Produkt- und Projektziele	Herr Müller Tel: 4711 Mail: M@oth.de	Kennt Vorgänger-produkte im Detail, da vorher selbst Anwender	5% verfügbar	Entscheidung über Realisierung, Geldgeber
Anwender: Koch	Ist Benutzer des Systems (nicht des Mobilteils)	Herr Becker Tel: 0815 Mail: B@oth.de	Experte der Speisenverbuchung	Urlaub: 1.1.-14.1. 20% verfügbar	Systemanwender muss künftig Speisen verbuchen
...	...	...	...	...	...

Abbildung 4: Stakeholderliste

→ Übersehen von Stakeholdern kann zu unvollständigen Anforderungen führen

Checkliste zum Finden von Stakeholdern

- Wartungs- und Servicepersonal
  - Wartung und Service muss unkompliziert und zügig durchzuführen sein
  - Wichtig bei hohen Stückzahlen
- Produktbeseitiger
  - Wichtig, wenn ausgeliefertes Produkt nicht nur Software umfasst, Frage der Beseitigung (z.B. Umweltschutz), kann enormen Einfluss auf Zielsetzung von Produktentwicklung haben
- Schulungs- und Trainingspersonal
  - Liefern konkrete Anforderungen zu Bedienbarkeit, Vermittelbarkeit, Hilfesystem, Dokumentation, Erlernbarkeit
- Marketing und Vertriebsabteilung
  - Als interne Repräsentanten externer Kundenwünsche und Marktentwicklung
- Systemschützer
  - Stellen Anforderungen zum Schutz vor Fehlverhalten von Stakeholdern
- Standards und Gesetze
  - Vorhandene und zukünftige Standards/Gesetze berücksichtigen

- Projekt- und Produktgegner
  - Vor allem zu Beginn des Projekts möglichst mit einbeziehen, sonst drohen Konflikte
- Kulturkreis
  - Setzt Rahmenbedingungen, z.B. verwendete Symbolik, Begriffe, ...
- Meinungsführer und die öffentliche Meinung
  - Beeinflussen oder schreiben Ziele vor, Zielmärkte berücksichtigen

4.7 Ermittlungstechniken

Befragungstechniken

- **Interviews**: Direktes Gespräch zwischen Requirements Engineer und Stakeholder.
- **Fragebögen**: Standardisierte Fragen zur Sammlung von Anforderungen.

Dokumentengetrieben

- **Systemarchäologie**: Analyse der Dokumentation bestehender Systeme.
- **Perspektivenbasiertes Lesen**: Untersuchen von Dokumenten aus verschiedenen Perspektiven.
- **Wiederverwendung**: Bereits erstellte Anforderungen nutzen.

Beobachtungstechniken

- **Feldbeobachtung**: Beobachtung des Stakeholders in dessen gewohnter Umgebung.
- **Apprenticing**: Requirements Engineer erlernt die Tätigkeit des Stakeholders.

Kreativitätstechniken

- **Brainstorming**: Ideenfindung in Gruppen.
- **Perspektivenwechsel**: Betrachtung des Problems aus verschiedenen Blickwinkeln.

4.8 Anforderungen dokumentieren

Benutzeranforderungen (Was? und Wofür?):

- Aussagen in natürlicher Sprache (Vorteile/Nachteile?)
- Einfache Diagramme (Vorteile/Nachteile?)
- Beschreiben Dienste, die System leisten soll
- Beschreiben Randbedingungen unter denen System betrieben wird
- Systembeschreibung aus Kundensicht **(→ Lastenheft)**

Systemanforderungen (Wie? und Womit?):

- Detaillierte Festlegung von Funktionen, Diensten und Beschränkungen
- Beschreibung, was implementiert werden soll
- Systembeschreibung aus technischer Sicht **(→ Pflichtenheft))**

4.8.1 Lastenheft

- Wird von Auftraggeber erstellt
- Beschreibung des **“Was”** und **"Wofür”**
- “Grobes” Pflichtenheft
- Details werden bewusst offen gelassen

4.8.2 Pflichtenheft

- Wird vom Auftragnehmer erstellt
- Beschreibung des **Wie** und **“Womit”**
- Zu lieferndes System wird detailliert
- Systembeschreibung aus technischer Sicht

Templates nach Balzert:

Lastenheft

1. Vision und Ziele
2. Rahmenbedingungen
3. Kontext und Überblick
4. Funktionale Anforderungen
5. Qualitätsanforderungen

Pflichtenheft

1. Vision und Ziele
2. Rahmenbedingungen
3. Kontext und Überblick
4. Funktionale Anforderungen
5. Qualitätsanforderungen
6. Abnahmekriterien
7. Subsystemstruktur (optional)
8. Glossar

→ *das Pflichtenheft ist eine Verfeinerung des Lastenheftes*

4.9 Anforderungen prüfen und abstimmen

Prüfung

- **Inspektionen**: Systematische Überprüfung der Anforderungen.
- **Reviews**: Überprüfung durch Stakeholder und Fachexperten.
- **Walkthroughs**: Schritt-für-Schritt-Durchgang der Anforderungen.

Validierung

- Überprüfung jeder Anforderung gegen Visionen und Ziele.
- Stakeholder-Review und formale Abnahme.

4.10 Kano-Modell

1. Basis-Merkmale:
   - Grundlegend, Kunden merken es bei Nichterfüllung (implizite Erwartungen)
   - Nichterfüllung führt zu Unzufriedenheit, Erfüllung führt nicht zu Zufriedenheit
   - Geringe Nutzensteigerung gegenüber Wettbewerbern
   - Beispiel Auto: Sicherheit, Rostschutz
2. Leistungs-Merkmale:
   - Kunden bewusst, beseitigen Unzufriedenheit oder schaffen Zufriedenheit
   - Beispiel Auto: Fahreigenschaften, Beschleunigung, Lebensdauer, Verbrauch
3. Begeisterungs-Merkmale:
   - Nutzen stiftend, unerwartet, zeichnen Produkt aus, rufen Begeisterung hervor
   - Kleine Leistungssteigerung, hoher Nutzen
   - Beispiel Auto: Sonderausstattung, besonderes Design, Hybridtechnologie
4. Unerhebliche Merkmale:
   - Ohne Belang für Kunden, stiften keine Zufriedenheit, führen nicht zu Unzufriedenheit
   - Beispiel Auto: Automatikgetriebe, Schiebedach (je nach Kundengruppe)
5. Rückweisungs-Merkmale:
   - Führen bei Vorhandensein zu Unzufriedenheit, bei Fehlen keine Zufriedenheit
   - Beispiel Auto: Rostflecken, abgelaufener TÜV

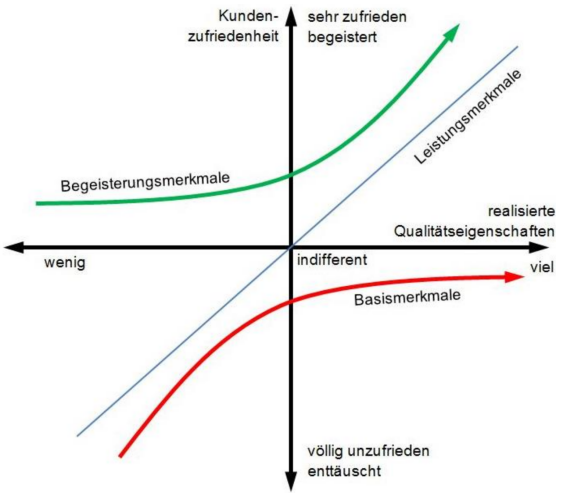


Abbildung 5: Veranschaulichung des Kano-Modells

Kano Modelle werden auf Basis von Kunden Befragungen ausgefüllt!
Hier ein Beispiel eines Kano Modells für intelligentes Krankenbett:

Wörtliche Kundenaussage	Kategorie	Zufriedenheit	Erfüllungsgrad	Aufwand/Nutzen	Priorität
Nie ist jemand erreichbar	Basis	-3	66%	↑	A
Der Patient muss die gewünschte Position schnell und einfach einstellen können	Basis	-1	75%	→	B
Das Bett muss beweglich sein und Türen passieren	Basis	0	100%		Erfüllt
Ich habe für die Reinigung täglich 2 Minuten Zeit	Leistung	3	70%	↑	A
Eine App, um den Zustand zu prüfen, wäre toll	Begeisterung	2	35%	↑	A
Ein Bett, das selbständig rund um die Uhr Vitalfunktionen überwacht	Begeisterung	4	45%	→	B
Ein selbstreinigendes Bett wäre eine echte Innovation	Begeisterung	5	80%	↓	C

Abbildung 6: Beispiel für Kano-Modell Tabelle

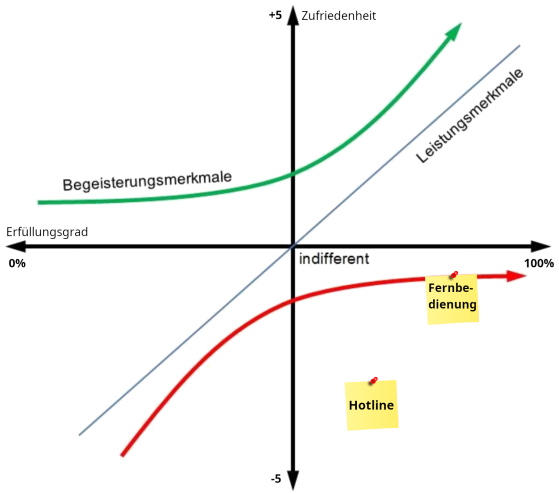


Abbildung 7: Beispiel für Kano-Modell Graph

Die nicht erreichbare Hotline zum Beispiel, wird von den Kunden als -3 in der Zufriedenheit und 66% im Erfüllungsgrad gewertet. Auf dem Graphen ist dieses Merkmal unter der Basismerkmal Kurve und somit sehr schlecht.
Dadurch ist der Aufwand/Nutzen hoch und die Priorität 'A'.

4.11 Best practices

- Kunden und Benutzer einbinden
- Alle möglichen RE-Quellen identifizieren und konsultieren
- Erfahrene Projektmanager und Teammitglieder einsetzen
- 15 – 30 % der Ressourcen für RE vorsehen
- Anforderungsschablonen und Beispiele zur Verfügung stellen
- Gute Beziehungen zu allen Beteiligten und Betroffenen herstellen
- Anforderungen priorisieren
- Ergänzende Modelle gemeinsam mit Prototypen entwickeln
- Eine Nachverfolgungsmatrix pflegen
- Peer Reviews durch Benutzer, Szenarien und Walk-Throughs zur Validierung und Verifizierung der Anforderungen durchführen

4.12 Erfassung von Anforderungen

4.12.1 Requirements Analysis

1. Erfassung der Systemaufgaben mit **Use Cases**
2. Beschreibung der Aufgaben mit **Aktivitätsdiagrammen**
3. (optional) Formalisierung der Beschreibungen in **Anforderungen**
4. Aufbau eines tieferen Verständnisses durch **Klassenmodellierung** und **Sequenzdiagramme** (Grobdesign)

→ **Was** sind die Hauptaufgaben des Systems?
→ **Wer** ist an den Aufgaben beteiligt?
→ **Welche Schritte** gehören zur Aufgabenerfüllung?

4.13 Use Cases

Ziel und Definition: Ein Use Case beschreibt eine konsistente und zielgerichtete Interaktion eines Benutzers mit dem System aus Anwendersicht. Er legt fest, was das System leisten muss, aber nicht wie.
Hauptaspekte:

- **Erfassung der Systemaufgaben:**
  - Erfassung, Beschreibung und Formalisierung der Aufgaben.
  - Iterativer Prozess mit Klassenmodellierung und Sequenzdiagrammen.

Erfragung des "WAS"?

- Was sind die Hauptaufgaben des Systems?
- Wer ist beteiligt?
- Welche Schritte gehören zur Aufgabenerfüllung?
- Aufgaben als Use Cases, Beteiligte als Akteure.

Struktur eines Use Cases:

- Titel, Kurzbeschreibung, Aktoren, Vorbedingungen, Ablauf, Auswirkungen, Anmerkungen.

Akteure: Ein Akteur ist ein Objekt der Systemumgebung, das mit dem System interagiert und Use Cases auslösen kann. Akteure können Personen, externe Geräte oder Nachbarsysteme sein.
Anmerkungen zu Use Cases:

- Sinnvolle Bezeichnung mit Substantiv und Verb.
- Fachlichen Auslöser und Ergebnis definieren.
- Abstrakt wie möglich, konkret wie nötig formulieren.

Use-Case-Diagramme (Standardelemente):

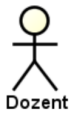
- Aktor, Use Case mit Bezeichner [Name], Systemgrenze mit Bezeichner [System], Kommunikationsbeziehungen, Beziehungen zwischen Anwendungsfällen.

Analyse von Use Case Dokumentationen:

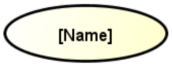
- Identische Abläufe als eigene Use Cases ausgliedern (**«include»-Stereotyp**).
- **«extend»-Stereotyp:** Variation des Use Cases, ausgeführt unter bestimmten Bedingungen (z.B. Sonderfall, Fehlerbehandlung).

4.13.1 Standardelemente

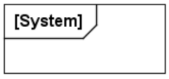
Aktor



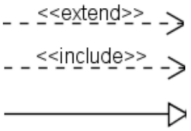
Use Case



Systemgrenze



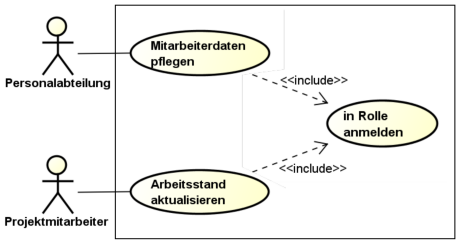
Kommunikationsbeziehung zwischen Aktoren und Use Cases



Beziehung zwischen Anwendungsfällen

Beispiele

«include»-Stereotyp



«extend»-Stereotyp