Software-Engineering – Teil 2, SS

2. Anforderungsanalyse bis Projektvertrag

Prof. Dr. Eckhard Kruse
DHBW Mannheim

Projektziel

Ein klar definiertes Ziel ist Voraussetzung für ein erfolgreiches Projekt:

- Was genau soll erreicht werden? → Ergebnisse/Deliverables
- Warum? Worin besteht der Nutzen? → Business Case
- Wer ist der Kunde, der Anwender, wer erhält die Ergebnisse?
- Überprüfungskriterien: Wie kann erkannt/gemessen werden, dass die Ziele erreicht worden sind?

- *"Der Weg ist das Ziel" gilt nicht für Projekte.*
- *Eher: "Kein Ziel => kein Weg => kein (erfolgreiches) Projekt"*
- *Viele Menschen sind zielstrebig …*

  … doch die meisten streben zu viel und zielen zu wenig.

Ziele sollen SMART sein

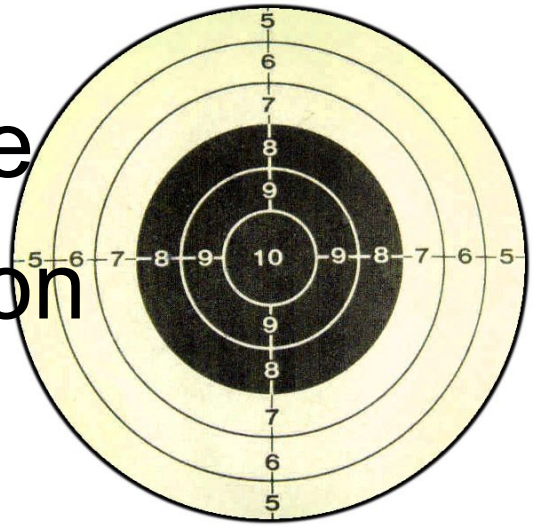
Specific

Measurable

Agreed upon

Realistic

Time-limited



Spezifisch

Messbar

Akzeptiert

Realistisch

Terminiert

(andere Auslegung: Achievable + Relevant ...)

Ideen

These:

Wenn Studenten nach einer halben Stunde Nachdenken endlich mal eine halbwegs gute Idee haben, sind sie so froh darüber, dass sie sich sofort begeistert in die mehrmonatige Umsetzung stürzen ...

… anstatt einige zusätzliche Stunden zu investieren, um wirklich gute Ideen zu finden.

Ist das effizient? (Wirkung / Zeiteinsatz)

(Gilt nicht nur für Studenten, sondern für die meisten Menschen, schauen Sie sich mal in Beruf, Alltag und Politik um!)

Ideenfindung

Kreativitätsforschung: Große Ideen entstehen nicht beim Nachdenken!

Typischer Ideenprozess (kennen Sie das?):

1. Bearbeitung des Problems, analysieren, alle Seiten betrachten, brainstormen, grübeln …
2. Loslassen: Joggen gehen, entpannen, ruhen, essen, duschen, Zähne putzen, einfache manuelle Tätigkeit o.ä.
   (Facebook + Internetsurfen gehören nicht dazu)
3. Heureka (Ich hab's)! Plötzlich ist die neue, große Idee da.
4. Ausarbeiten, verfeinern, anwenden...

Auch:
1. Vorbereitung - 2. Inkubation - 3. Inspiration - 4. Gestaltung

Ideen erzeugen

Ist-Analyse: Erhebung von Anforderungen auf Basis eines existierenden Systems und bisheriger Erfahrungen (ggf. gezielt: Schwachstellenanalyse)

Kreative Techniken dienen zur Erzeugung neuer Ideen und Lösungsansätze. (Bei Entwicklung neuer Systeme unumgänglich)

- Brainstorming
  - freies Fantasieren und Assoziieren. Ziel: Möglichst viele Ideen (Quantität vor Qualität)
  - Erst in einem zweiten Schritt wird bewertet, gruppiert, gefiltert.
- Methode 635 ("Brainwriting"-Technik)
  - 6 Teilnehmer schreiben 3 Ideen in 5 Minuten auf
  - Blätter (insgesamt 5 mal) weiterreichen: Ideen weiterentwickeln
- Mind maps / concept maps
  - Konzepte/Ideen als Knoten und Beziehungen als Pfeile
  - auf Papier (Mind mapping Programme bremsen und engen ein.)
- Experimenteller...: Semantische Intuition, “Kopfstand“
- Weitere Kreativitätstechniken:
  http://de.wikipedia.org/wiki/Kreativit%C3%A4tstechnik#Methoden

Wichtig: Ideen schriftlich/grafisch festhalten! (ggf. Moderator, Protokollant getrennt)

Ideen entwickeln: Grafisch

Brain Sketching (grafische Alternative zu Brainwriting)

- typisch: 4-8 Teilnehmer
- Vereinbarung, welche Frage addressiert werden soll
- Die Teilnehmer erstellen auf separaten Blättern, unabhängig voneinander Skizzen.
- Blätter werden weitergereicht, ergänzt, mit Kommentaren versehen usw.

Skizzierung (von Teilen) der Benutzeroberfläche

- Layout, einzelne Sichten
- Stift und Papier
- Vorläufer zum Mockup

Storyboarding

- Anwendungsszenarien und Fälle als Comic erzählen.
- Darstellung von Anwendern und System, Interaktion, Gedanken
- Darstellung von UI-Teilen und Interaktionen (z.B. Pfeile wie Aufmerksamkeit / Interaktion abläuft)

Ideen entwickeln

Teamarbeit

2.1 Ideen entwickeln

Bzgl. der zu entwickelnden Lösung hat der Kunde noch viele Freiheiten gelassen. Entwickeln Sie Ideen:

a) Welches Problem soll angegangen werden: Projektzielsetzung? Produktfeatures? Anwendungsszenarien? Name? ...

b) Setzen Sie strukturierte Brainstorming-Techniken ein (z.B. Mindmaps, 6-5-3-Brainwriting, Semantische Intuition...).
   - Wieviel Zeit? Protokollant? Moderator?
   - Schreiben Sie die Ergebnisse auf.

c) Wie könnte das Ganze aussehen? → Skizzen/grafische Verfahren
   (Tipp: Papier und Stifte sind für Brainstorming meistens die besten Werkzeuge, SW-Tools lenken mitunter ab / bremsen die Gedanken aus.)

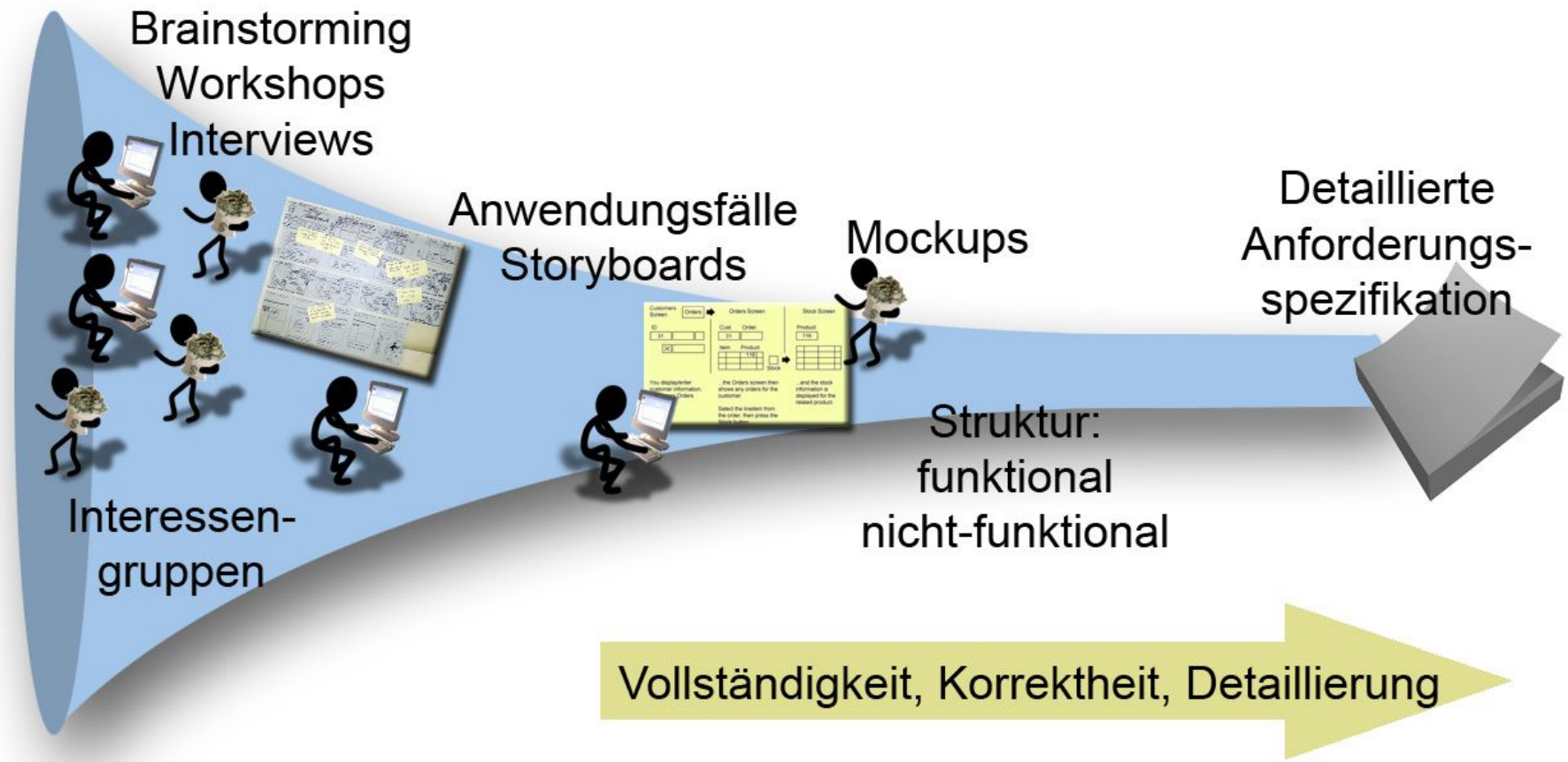
d) Erstes Filtern/Priorisieren:
   - Was sollte unbedingt festgehalten werden?
   - Was ist noch näher auf seinen Wert hin anzuschauen?
   - Was wird (erst mal) zur Seite gelegt? (wo dokumentiert...?)

Anforderungen / Requirements

Die **Anforderungsanalyse (requirements analysis)** dient der Erhebung, Untersuchung und Bewertung aller Anforderungen des Auftraggebers an die gewünschten Projektergebnisse.

- Verwandte (oft synonym verwendete) Begriffe:
  - Anforderungserhebung (requirements elicitation, requirements engineering)
  - (Anforderungs)-Spezifikation (requirements specification)
- Anforderungen aus Auftraggebersicht: Lastenheft (z.B. bei Ausschreibungen)
- Anforderungen aus Auftragnehmersicht: Pflichtenheft (ist detaillierter als Lastenheft und berücksichtigt bereits Lösungsansätze)
- Das Pflichtenheft ist Vertragsbestandteil
- Unterscheidung: funktionale und nicht-funktionale Anforderungen
- Zentrale Frage: Wie genau müssen die Anforderungen zu Vertragsabschluss erfasst sein? Was kann nachher noch ausgearbeitet werden?

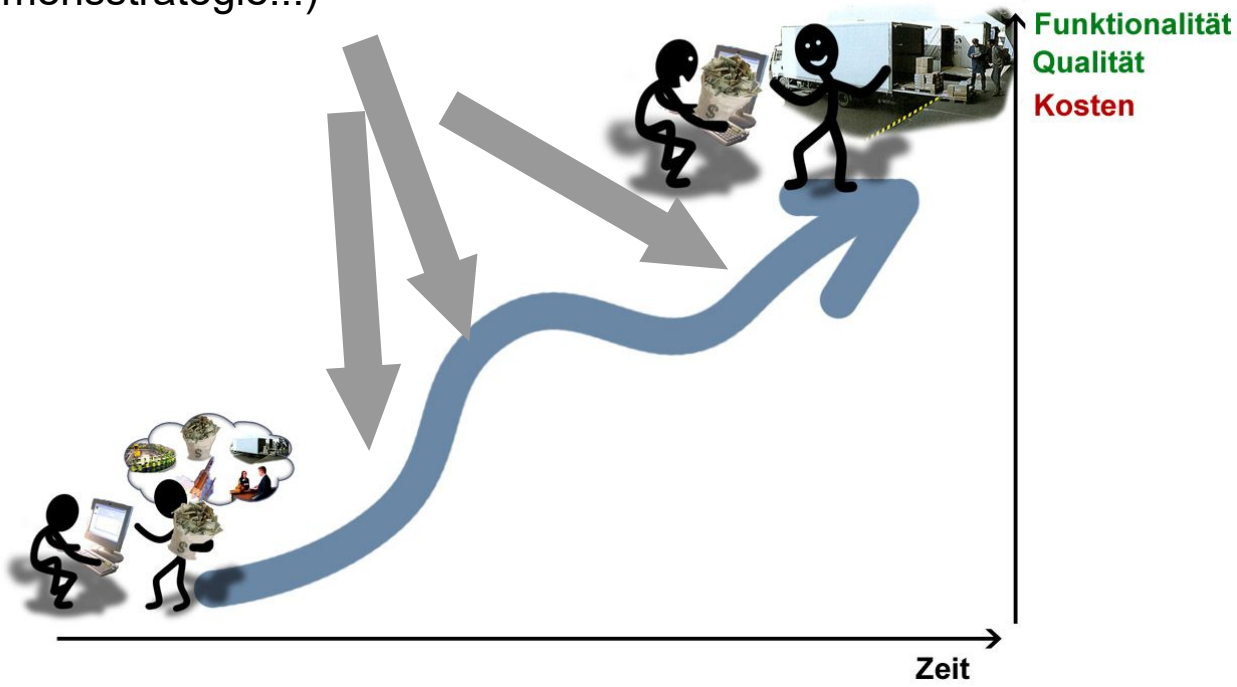
Anforderungsanalyse Vorgehen



Anforderungsmanagement

Anforderungen können sich über die Projektlaufzeit und auch nach Release des Produktes verändern:

- Fehler bei der Anforderungsanalyse werden erkannt
- Neue, unvorhergesehene Kundenwünsche
- Anforderungen von außen (neue Standards, Änderung der Produkt-/Unternehmensstrategie...)



Anforderungsmanagement

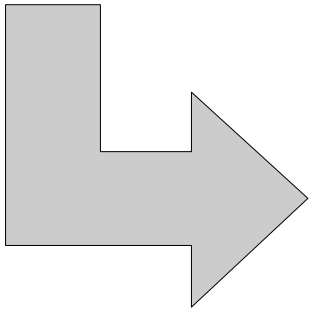
Anforderungsmanagement (requirements management) betrachtet das Erheben und Verwalten von Anforderungen als einen Prozess, der über die gesamte Projektlaufzeit gesteuert werden muss.

- Ermitteln von Anforderungen
- Analyse von Anforderungen
- Beschreiben von Anforderungen → Pflichtenheft
- Priorisieren von Anforderungen
- Prüfen von Anforderungen → Testen/Validieren
- Verwalten von Anforderungen
- Nachverfolgung von Anforderungen (Requirements traceability)
- Ändern von Anforderungen → Change Management

Anforderungsanalyse - Lessons learned?

Anforderungsanalyse + Anforderungsmanagement
Rückblick WS-Projekt – Ausblick SS-Projekt

- Was lief gut im Wintersemester?
- Was möchten Sie wieder auf ähnliche Weise machen?
- Was lief nicht so gut?
- Was möchten Sie diesmal anders/besser machen?
- Wie können Sie noch effizienter werden?



- Ermitteln von Anforderungen
- Analyse von Anforderungen
- Beschreiben von Anforderungen → Pflichtenheft
- Priorisieren von Anforderungen
- Prüfen von Anforderungen → Testen/Validieren
- Verwalten von Anforderungen
- Nachverfolgung von Anforderungen (Requirements traceability)
- Ändern von Anforderungen → Change Management

Anforderungsmanagement - Werkzeuge

SW-Tools helfen bei der Verwaltung von Anforderungen:

- Zentrale Verwaltung aller Anforderungen:
  - Welche Anforderung, von wem gefordert, Priorität, Änderungshistorie...
- Abhängigkeiten von Anforderungen untereinander
- Suche, verschiedene Sichten, gezielte Navigation
- Impact-Analyse: Wie / auf wen wirken sich Anforderungsänderungen aus
- Eindeutigkeit, Konsistenzchecks
- Traceability: Verursacher, Änderungen = Lifecycle einer Anforderung
- Versionierung, Status (z.B. proposed vs. accepted), Freigaben
- Referenzen auf andere Artefakte/Werkzeuge (z.B. UML Use Case Diagramme)
- Zuordnung von Aufwandsschätzungen
- ...

Wie verwalten Sie Ihre Anforderungen im Projekt?

Warum ist das Pflichtenheft (in realen Projekten) nicht genug?

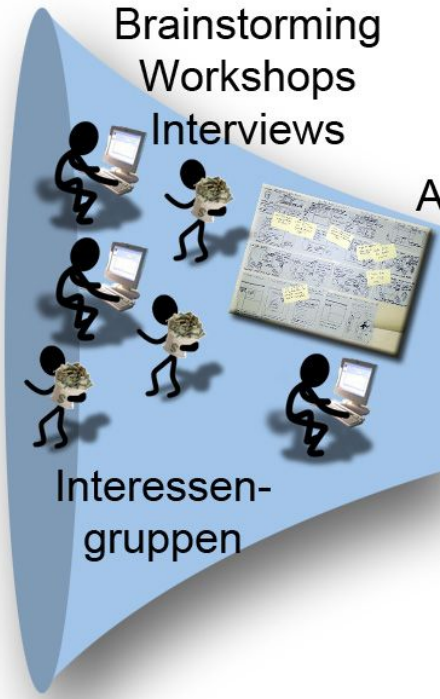
Zahlreiche kommerzielle Produkte...

- s. z.B.: http://makingofsoftware.com/resources/list-of-rm-tools
- Evtl. Minimalvariante: Tabellen (z.B. Excel) + Dokumente

Ermittlung von Anforderungen

Wichtig: Systematisches, umfassendes Vorgehen:

- Verschiedene Interessengruppen (stakeholders) einbeziehen:
  - z.B. Anwender, Administrator, Manager ...
  - Interviews, Diskussionen
- Verschiedene Sichten (Viewpoints) berücksichtigen:
  - Anwendersicht: z.B. einfache Bedienbarkeit
  - Administrator: z.B. Sicherheit, einfache Installation, Verwaltung
  - Manager: z.B. Total cost of ownership
  - …
- Umfeldanalyse:
  - Andere Produkte auf diesem Markt, Wettbewerber?
  - Trends (Technologien, Benutzererwartungen)
  - Richtlinien und Standards (im Unternehmen, im Marktumfeld



Benennen Sie Interessengruppen, Sichten, Umfeldfaktoren (im Pflichtenheft)!

Ist-Analyse:
Bestehendes Wissen auswerten

→

Kreativitätstechniken:
Neue Ideen erzeugen

→

Ideen entwickeln, ausarbeiten, dokumentieren

Anforderungen: Ist-Analyse

Eine **Ist-Analyse** dient der Bestandsaufnahme des aktuellen Zustands, bisheriger Erfahrungen, bekannter Defizite, vorhandener Wünsche usw.

Verschiedene Methoden:

- Interviews (Frage/Antwort mit Anwendern / dem Auftraggeber)
  - Vorteile: Interaktiv, flexibel, dadurch umfassender
  - Nachteile: Zeitaufwändig, Terminfindung u.U. schwierig, erfordert kompetenten Interviewer
- Fragebogen / Checklisten (an Anwender/Auftraggeber)
  - Vorteile: Reproduzierbar, schriftliche Ergebnisse, viele Anwender mit wenig Aufwand erreichbar
  - Nachteile: evtl. geringe Antwortquote, Missverständnisse, starr, kaum Platz für neue Ideen → nur als Zusatzmaßnahme geeignet
- Dokumentenanalyse (durch den Auftragnehmer)
  - Vorteile: Kaum Aufwand für Auftraggeber
  - Nachteile: meist unvollständig, viel undokumentiertes Wissen, kein direkter Austausch mit Anwender
- Beobachtung (des Anwenders bei der Arbeit)
  - vor allem ergänzend für Usability-Anforderungen

Ist-Situation + Motivation
→ Pflichtenheft

Machbarkeitsstudie

Um Fehlinvestitionen zu verhindern und Risiken zu minimieren, kann bei Zweifeln an dem Erfolg eines geplanten Projektes eine **Machbarkeitsstudie** vorangestellt werden.

Eine Machbarkeitsstudie soll Fragen zum geplanten Projekt klären, z.B. in Bezug auf:

- Wirtschaftlichkeit
- Technische Machbarkeit
- Marktsituation, Unternehmens- und Produktstrategie
- Patentrechtliche Fragen
- Aufwandsabschätzung
- Make or Buy Entscheidungen
- Grundlage für Angebotserstellungen

Wo spielen in Ihrem Projekt Überlegungen zur Machbarkeit eine Rolle? Wie gehen Sie damit um?

Ideen ausarbeiten

Definition

Usage Scenarios (Anwendungsszenarien) beschreiben die Interaktion (in einem größeren Kontext) von einer/mehreren Personen mit dem System.

- potenziell umfangreich, umfassen meist mehrere Use Cases.
- Verschiedene Detaillierung: grobe Schritte bis hin zu jedem einzelnen Mausklick

Szenario: "Der Kunde verwaltet sein Bankkonto über das Internetportal."

Use Cases:
"Anzeige des aktuellen Kontostands",
"Überweisung"
...

User Stories sind kurze (meist ein, zwei Sätze) umgangssprachliche Beschreibungen einer Anforderung aus Anwendersicht.

- weniger umfangreich und formal als Use Cases
- vor allem im XP/Agile-Umfeld, zur Aufwandsschätzung

"Der Anwender kann den aktuellen Zustand der Simulation speichern und laden."

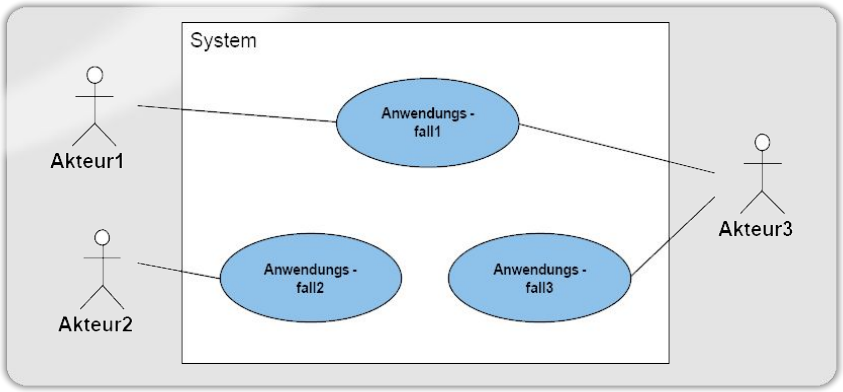
"Beim Schließen des Programms wird der Anwender gefragt, ob er seine Daten speichern möchte (sofern sie in einem verändertem Zustand sind).

Anwendungsfälle (Use Cases)

Definition

Ein **Anwendungsfall (Use case)** beschreibt einen typischen Vorgang aus mehreren zusammenhängenden Arbeitsschritten, die von einem **Akteur (Actor)** durchgeführt werden, um ein bestimmtes Ziel zu erreichen / eine Aufgabe zu erledigen.

Beispiel:



#15 Adding a User
A new user needs an account to access the system.
A new user is added to the system by creating an account consisting of a user name, password, and home directory.
Priority: 5 Estimation: Easy

#16 Logging onto the System
A user logs onto the system by entering their username and password. If they enter an incorrect value for either field, they are provided an error message indicating. if they enter three incorrect values, their account is locked and the user must inform the IT department to restore their account.
Priority: Medium Estimation: 3

#135 Paying for a Book
A customer may pay for a book using a credit card (VISA,AMEX,MC), PayPal, or by check.
Priority: 12 Estimation: 1

Use Case Dokumentation

Detaillierte Checkliste für die Dokumentation von Use Cases:

- **Name:** A brief, descriptive, and unique name for the use case.
- **Description:** A short description of what the use case does.
- **Actors:** A list of all the actors that interact with the use case.
- **Priority:** A short description of how important the use case is in the overall scope of the application. Knowing the priority of each use case lets you design the architecture accordingly.
- **Status:** Notes how complete (or incomplete) the development of the use case is.
- **Preconditions:** A list of conditions that must be true before the use case starts.
- **Postconditions:** A list of conditions that must be true after the use case is complete.
- **Use case interactions:** Identifies other use cases the use case interacts with or relies upon.
- **Flow of events:** A list of events that happen during the execution of the use case. This could also contain alternative paths.
- **Activity diagram:** An activity diagram or diagrams of the flow of events or some part of the flow of events.
- **Secondary scenarios:** If the flow of events contains only a primary scenario, then here secondary scenarios might also be documented.
- **User interface:** A simplified picture of the user interface for the use case. A prototype of the user interface helps the development team see if the design is on the right track.
- **Sequence diagrams:** Sequence diagrams of the different scenarios.
- **View of participating classes:** A diagram of all the classes whose instances work together to implement the use case.
- **Other requirements:** Other requirements might include quality attributes if you do not have a specific document for them.

Quelle: http://www.ibm.com/developerworks/wireless/library/wi-arch9/

Use Case Dokumentation

Beispiel

Beispiel: Detaillierte Dokumentation von Use Cases:

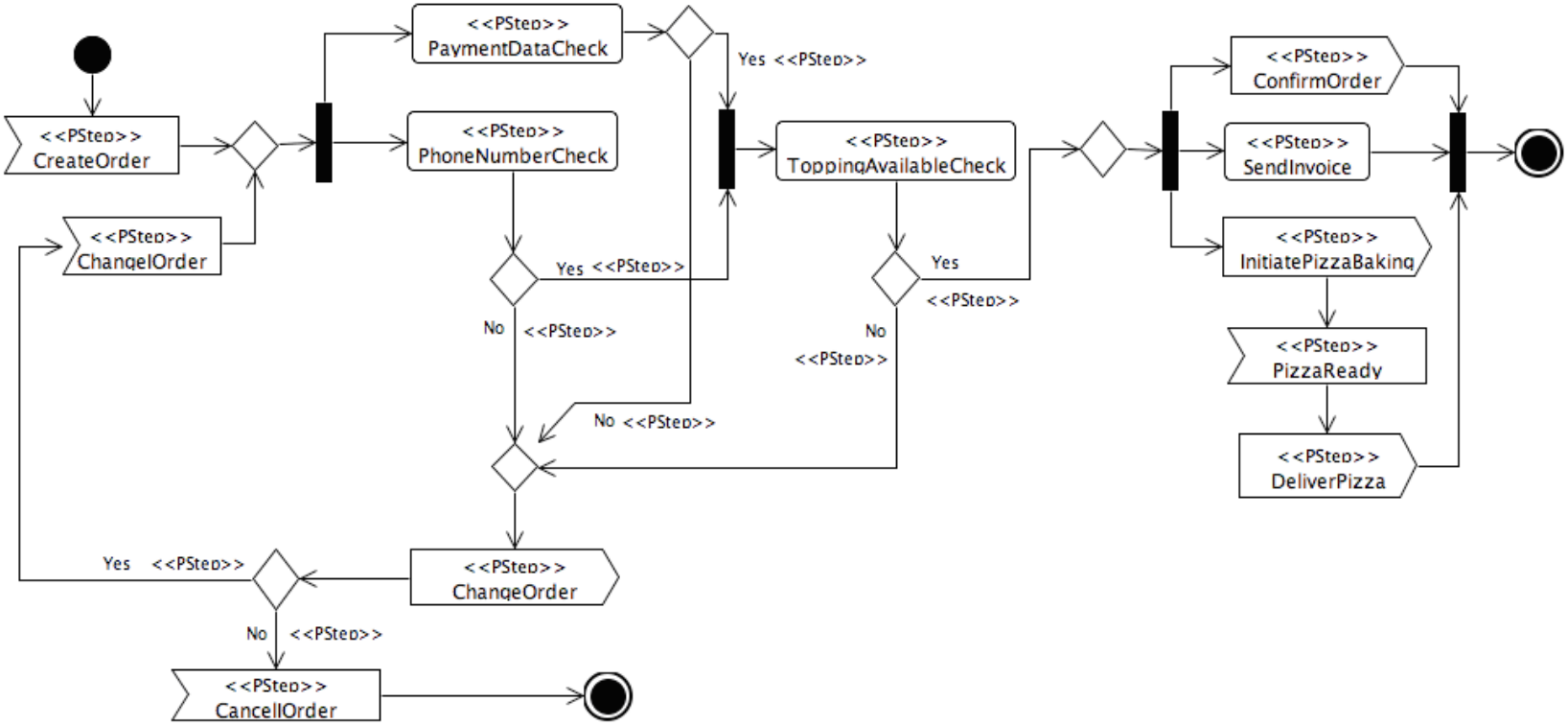
- **Name:** Login.
- **Description:** This use case handles the access to the system. If the user has no access (that is, is unregistered), he or she will be directed to a registration page.
- **Actors:** Customer, Customer Representative.
- **Priority:** Very important.
- **Status:** Started.
- **Preconditions:** The user has browsed to the Web store's front page and filled in his or her account details and must now log in.
- **Postconditions:** (1) If the user was in the system, he is logged in and has access to the system; (2) the main menu is shown to the user; (3) if the user was not in the system an error message is shown with a possibility to register for the site.
- **Use case interactions:** None.
- **Flow of events:**
  - basic path: (1) The use case starts when the user browses into the Web store's front page; (2) the system will display the front page with text boxes for username and password, a button for accepting the information, and a link for registration; (3) the user enters a username and password; (4) the system checks the information; (5) the system displays the front page with user-specific information; (6) the use case ends.
  - **Alternative path:** … (3) the user clicks the link to register; (4) the system displays a registration form with text boxes for name, address, telephone number, and e-mail; ...

Quelle: http://www.ibm.com/developerworks/wireless/library/wi-arch9/

UML in der Anforderungsanalyse

Beispiel

Aktivitätsdiagramm (activity diagram)

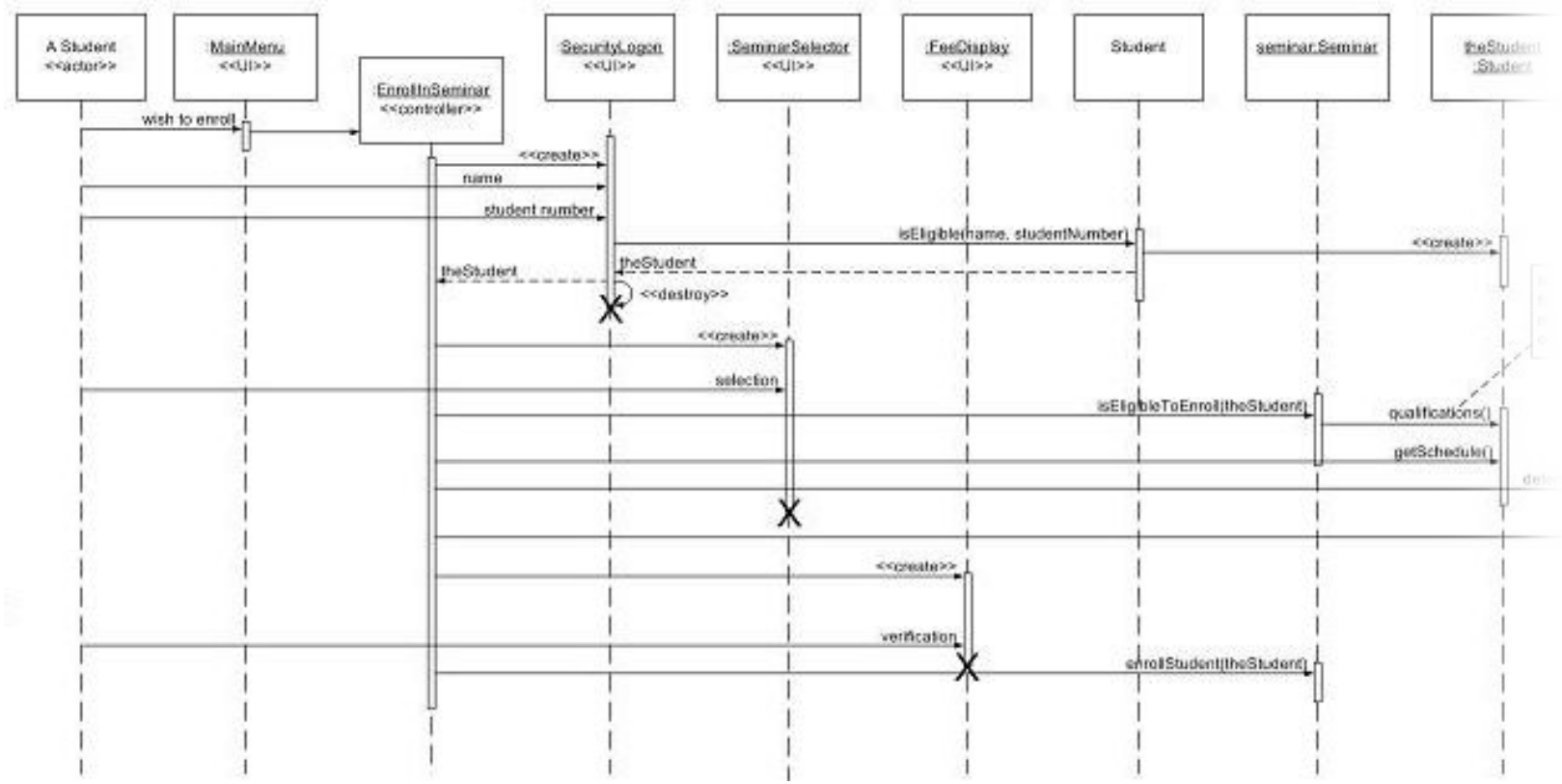


http://www.eclipse.org/m2m/atl/usecases/UML2AnyLogic/img/PizzaOrdering-UML2ActivityDiagram.PNG

UML in der Anforderungsanalys

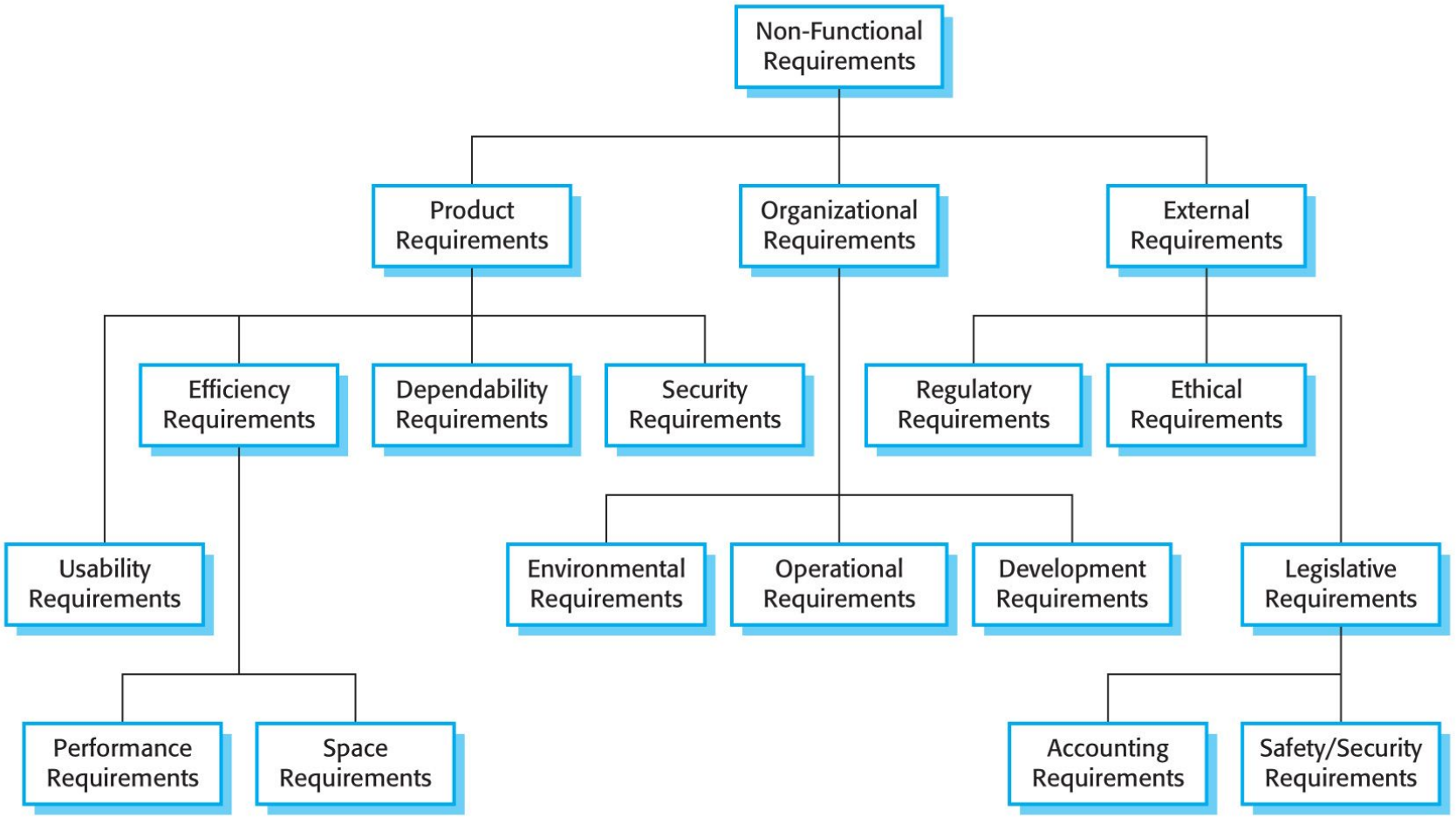
Beispiel

Sequenzdiagramm (sequence diagram)



http://www.agilemodeling.com/images/models/sequenceDiagramBasicCourse.jpg

Nicht-funktionale Anforderungen



Quelle: Ian Sommerville, Software-Engineering

Pflichtenheft

Definition

Das **Pflichtenheft** definiert die Anforderungen an das zu liefernde (Software-)Produkt. Basierend auf dem Lastenheft wird es vom Auftragnehmer in Abstimmung mit dem Auftraggeber durch Ausarbeitung weiterer Anforderungsdetails erstellt. Das Pflichtenheft ist verbindliche Grundlage des abzuschließenden Vertrages.

Beispielstruktur eines Pflichtenhefts (nach Balzert, Softwaretechnik)**:**

1. **Ziele:** Was muss das Produkt unbedingt leisten? Was ist wünschenswert? Was ist nicht Produktbestandteil?
2. **Einsatzbereich:** Anwendungsbereiche? Zielgruppen?
3. **Umgebung:** Welche Hardware- / Softwareumgebung muss unterstützt werden?
4. **Funktionen**: Detaillierte, systematische Auflistung aller wesentlichen Produktfunktionen
5. **Daten**: Detaillierte, systematische Auflistung aller wesentlichen zu handhabenden Daten
6. **Leistungsmerkmale**: Auflistung aller nicht-funktionalen Anforderungen (Performanz, Datenvolumen, Skalierbarkeit, Bedienbarkeit, Zuverlässigkeit usw.)
7. **Benutzeroberfläche:** Grundlegende Anforderungen an die Benutzeroberfläche?
8. **Qualitätsziele:** Wie wird die Produktqualität sichergestellt? Welche Standards/Normen werden befolgt?
9. **Testszenarien:** Anhand welcher Testfälle wird die Erfüllung der Anforderungen überprüft?
10. **Entwicklungsumgebung:** In welcher Umgebung / mit welchen Technologien wird die Software entwickelt?
11. **Ergänzungen:** Weitere wichtige Punkte, z.B. Lizenzen, Patente, 3rd Party Komponenten
12. **Glossar:** Definition von verwendeten Abkürzungen und Fachbegriffen

Anforderungsanalyse

Teamarbeit

2.2 Anforderungsanalyse

Nachdem Sie erste Ideen für das Projekt entwickelt haben, steigen Sie nun in die systematische Anforderungsanalyse ein:

a) Beschreiben Sie Anwendungsszenarien und wesentliche Use Cases

b) Sammeln und strukturieren Sie alle Anforderungen tabellarisch. Bilden Sie geeignete Kategorien.

c) Unterscheiden Sie funktionale und nicht-funktionale Anforderungen.

d) Planen Sie für die weitere Anforderungsanalyse Aktionen und ordnen Sie sie den Teammitgliedern zu (→ Aktivitätsliste).

e) Erläutern sie die Anforderungen schriftlich. Erstellen Sie dazu ein Dokument, das sich schrittweise zu einem Pflichtenheft entwickeln kann.

f) Welche Anforderungen erfordern detailliertere Vorstudien?

Erster grober Zeitplan?

- Teams, Themen, Kickoff
- Anforderungsanalyse, Projektplanung+Aufwandsschätzung (grob)
- Vorstudie(n) + Mockups
- Entwurf/Architektur
- Projektplanung+Aufwandsschätzung (fein)
- Moduldesign + Implementierung
- Testspezifikation + Testen, Validierung
- Benutzerdoku
- Übergabe + Präsentation
- Abgabe aller Materialien + Projekttagebücher

Wie sieht Ihr derzeitiger (grober) Zeitplan aus?

Validierung von Anforderungen

Anforderungen sollten abschließend überprüft werden:
(wie alle Ergebnisse im SW-Engineering-Prozesses)

- Validierung: *"Haben wir das Richtige spezifiziert?"*
  - Alle wesentlichen Bereiche addressiert?
  - Funktionale und nicht-funktionale Eigenschaften?
  - Ziele im Auge behalten!
- Verifikation: *"Haben wir richtig spezifiziert?"*
  - Sind die spezifizierte Anforderungen ausreichend für die Umsetzung bis hin zur Abnahme?
  - Lassen sich Testfälle auf Basis der Anforderungen definieren?
  - Welche Auswirkungen haben geänderte/neue Anforderungen?

Verfahren zum Überprüfen von Anforderungen:

- Reviews oder Walk-throughs
- Audits oder Inspektionen
- Prototyping

Anschließend: Freigabe + Einfrieren als "Baseline" = Menge der Anforderungen als verbindlicher Vertrag. Anschließende Änderungen nur noch über definierten Change Management Prozess.

Anforderungen → Testfälle

Ob die Anforderungen erfüllt werden, zeigt sich letztlich (hoffentlich) beim Testen:

- Sind die richtigen Anforderungen spezifiziert worden?
- Welche Anforderungen sind getestet und wie ist das Testergebnis?
- Für welche Anforderungen gab es negative Testergebnisse? Konsequenzen?
- Akzeptanztest: Erfüllt das System aus Anwendersicht seinen Zweck (ggf. werden hier auch fehlende/falsche Anforderungen erkannt.)

→ Hilfreich: Frühzeitig Testfälle definieren, mit denen gezielt die Anforderungen vollständig überprüft werden können. ("Requirements driven testing")

Anforderungsanalyse - Kundengespräch

Teamarbeit

2.3 Anforderungsanalyse - Kundengespräch

Im Rahmen der Anforderungsanalyse haben Sie einen Termin mit dem Kunden, um verbleibende Fragen bis zum Abschluss des Pflichtenheftes zu klären.

a) Wie ist der aktuelle Stand Ihrer Arbeiten?

b) Welche Punkte sind aus Ihrer Sicht jetzt noch zu klären?

c) Was kann erst später entschieden werden? (z.B. abhängig von Vorstudien, Prototypen oder Aufwandsschätzungen)

d) Priorisierung der Anforderungen?

e) Nächste Schritte?

Aufwandsschätzungen

Voraussetzungen für die Fertigstellung des Pflichtenheftes:

- Detaillierte Anforderungen
  → Was möchte der Kunde? Was versprechen wir ihm?
- Grobes Vorgehen und Aufwandsschätzung
  → Können wir das leisten (und dabei wirtschaftlich arbeiten)?

Schätzverfahren

Frage: Wie groß ist der Aufwand/die Kosten für die Entwicklung des Gesamtsystems bzw. die Umsetzung einzelner Anforderungen?
(Vor allem Personalaufwand, da Hauptkostenfaktor)

Antwort: Im Projektmanagement und Software Engineering gibt es zahlreiche Verfahren um Aufwände zu schätzen.

Probleme: Qualität der Schätzung ist zu einem Großteil bestimmt durch:

- Erfahrung und Vorwissen der beteiligten Personen
- Granularität/Detaillierung der zu schätzenden Aufgaben

Schätzverfahren:

- Expertenbefragung
- Delphi-Methode
- Analogiemethode
- Relationsmethode
- Multiplikatormethode
- Gewichtungsmethode
- Prozentsatzmethode
- Function / Data / Object Point Methode
- ...

Expertenbefragung / Delphi-Methode

Ansatz: Systematische Befragung eines/mehrerer Experten.

z.B. unabhängig/anonym (Delphi-Methode), Diskussionsrunde, Schätzklausur, Mittelwerte aus Einzelschätzungen ...

Delphi-Methode:

1. Der Projektleiter schildert den Experten das Vorhaben und händigt Schätzformulare aus.
2. Die Experte füllen unabhängig das Formular aus. Fragen dürfen lediglich mit dem Projektleiter besprochen werden.
3. Der Projektleiter analysiert die Angaben. Weichen Schätzwerte eines Paketes stark voneinander ab, werden diese mit Kommentar auf einem neuen Formular erfasst.
4. Das neue Formular wird erneut zur selbständigen Überarbeitung an die Experten gereicht.
5. Schritte 2-4 werden ggf. wiederholt, bis die Ergebnisse (für den Projektleiter) stimmig sind und sich hinreichend angenähert haben.
6. Die letzte Überarbeitung (Mittelwerte aller Teilnehmer) wird als endgültiges Schätzergebnis genommen.

Varianten:

- Computer-gestützt. "Echtzeit-Delphi"
- Breitband-Delphi: Diskussionsrunden zwischen den Iterationen

Function point analysis

Die **Function point analysis (Funktionspunktanalyse)** teilt die Gesamtaufgabe in elementare Funktionseinheiten, welche die Grundlage der Aufwandsschätzung bilden.

1. Auflisten aller Funktionspunkte:
   - Eingaben, Ausgaben, Abfragen, Datenbestände, Referenzen, Logik, …
2. Gewichtungsfaktoren nach festem Schema unter Berücksichtigung von Schwierigkeitsgraden, z.B.
   - Eingaben: einfach: 3, mittel: 4, schwierig: 5
   - Datenbestände: einfach: 7, mittel: 10, schwierig: 15
   - ...
3. Gesamtsumme aller Funktionspunkte berechnen.
4. Allgemeinen Schwierigkeitsfaktor bestimmen, um allgemeine Systemmerkmale und nicht-funktionale Anforderungen zu berücksichtigen:
   - GUI-Gestaltung, Wiederverwendung, verteilte Anwendung, Performanz ...
5. Allgemeinen Schwierigkeitsfaktor bestimmen, um allgemeine Systemmerkmale und nicht-funktionale Anforderungen zu berücksichtigen:
6. Aufwand = Schwierigkeit*Summe Funktionspunkte
7. Zeitplanung: Entwicklung der Funktionspunkte über der Zeitachse auftragen.

Analogiemethode u.a.

Die **Analogiemethode** verwendet Erfahrungswerte aus vorherigen ähnlichen Entwicklungsprojekten/Arbeitsschritten als Grundlage für die Gewinnung der Schätzwerte.

- Voraussetzung: Die Schätzenden sollten eigene Erfahrungen aus zuvor selbst durchgeführten Projekten haben.
- Die Relationsmethode erweitert und formalisiert die Analogiemethode:
  - Berücksichtigung unternehmensspezifischer Faktoren, Erfahrung der Entwickler, Zielumgebung, Programmiersprache (in Relation zu Durchschnittswerten)
- Multiplikator-Methode:
  - erwartete Lines of Code (LOC) auf Basis bisheriger Erfahrungen
  - Faktoren für den Schwierigkeitsgrad
- Gewichtungs-Methode:
  - Multiplikator-Methode ergänzt um Faktoren, wie z.B. Programmiersprache, Mitarbeiterqualifikation

Aufwandsschätzung

Teamarbeit

2.4 Aufwandsschätzung

Führen Sie eine systematische Aufwandsschätzung durch:

a) Ist die Gesamtfunktionalität hinreichend fein strukturiert?

b) Führen Sie Schätzungen für alle Teile durch und berechnen Sie die Summe (Verwenden Sie z.B. eine Tabellenkalkulation zur Dokumentation und Berechnung).

c) Berücksichtigen Sie verschiedene Lösungsvarianten, bzw. unterschiedliche Einbeziehung der Anforderungen der Prios 1, 2, 3.

d) Beziehen Sie sich bei Ihren Schätzungen im Sinne der Analogiemethode auf Vergleiche aus dem Wintersemester. Gibt es Sonderfaktoren für erhöhte Schwierigkeiten?

e) Für welche Bereiche ist noch keine Aufwandsschätzung möglich? (z.B. wegen großer technischer Unsicherheiten oder verschiedener Alternativen, wie das System aussehen soll)

f) Definieren Sie Maßnahmen / Studien, um offene Punkte zu klären.

Prototyp

Ein **Prototyp** ist die rudimentäre, ansatzweise Implementierung (von Teilen) des zu entwickelnden Softwareprodukts. Im Vordergrund bei der Prototypentwicklung steht, möglichst schnell einen ersten Eindruck des Produktes zu erhalten - ohne Anspruch auf Vollständigkeit, hohe Software-Qualität oder gute Softwarearchitektur.

Einsatzbereiche eines Prototyps (insbesondere in den frühen Projektphasen):

- Demonstration (z.B. der Benutzerschnittstelle) bei den Interessengruppen des Projektes
- Ausprobieren neuer Ideen – z.B. im Rahmen der Anforderungsanalyse
- Aufwandsabschätzung, wie komplex bestimmte Entwicklungsaufgaben werden können
- Evaluation verschiedener technischer Lösungsansätze

Die eigentliche Produktentwicklung sollte (in aller Regel) nicht auf Basis des Prototyp-Codes erfolgen, sondern mit einem neuen, sauberen Entwurf!

Mock-up

Ein **Mock-up** ist ein Vorführmuster, um das Aussehen des zu entwickelnden Produktes zu demonstrieren (allerdings ohne Funktionalität).

- Einsatzbereich insbesondere in der Anforderungsanalyse, um Kunden/Anwender-Feedback zu erhalten.
- Mock-ups von Softwareprodukten stellen die Benutzerschnittstelle dar (= den sichtbaren Teil der Software) und können z.B. einfach mit Zeichenprogrammen o.ä. realisiert werden.
- Ein Mock-up ist kein Prototyp (denn ein Prototyp umfasst immer auch tatsächlich implementierte Funktionalität).

Vorstudien

Teamarbeit

2.5 Vorstudien

Definieren Sie Maßnahmen und Aktionen, um unklare Punkte vor Abschluss des Pflichtenheftes zu klären, z.B.

a) Mock Ups, um einen besseren Eindruck der Benutzerschnittstelle und Interaktionsmechanismen zu bekommen.

b) Recherchen, um Informationen zu Simulationsverfahren zu erhalten, die für das Projekt geeignet sein könnten.

c) Prototypen, um die Machbarkeit und den technischen Aufwand für bestimmte Realisierungsansätze abschätzen zu können.

d) Dokumentieren Sie die Ergebnisse der Vorstudien schriftlich (knappe Zusammenfassung: Fragestellung + Ergebnis, Empfehlung für das Projekt).

e) Aktualisieren Sie die Aufwandsschätzung.

(Addressieren Sie nur die wirklich notwendigen Dinge, um Risiken zu reduzieren - nehmen Sie nicht zukünftige Entwurfsarbeiten vorweg.)

Fertigstellung Pflichtenheft

Teamarbeit

2.6 Fertigstellung Pflichtenheft

Schließen Sie die Anforderungsanalyse ab und stellen Sie das Pflichtenheft fertig.

a) Haben Sie alle Anforderungen erfasst, angemessen strukturiert, priorisiert?

b) Haben Sie die Aufwände geschätzt? Ist das Projektvorhaben realistisch?

c) Stellen Sie das Pflichtenheft fertig. Ist es als Vertragsgrundlage geeignet?

d) Gibt es evtl. noch Punkte, die erst nach Vertragsabschluss, z.B. durch weitere Vorstudien, Analysen zu klären sind?

e) Definieren Sie den weiteren Prozess bzgl. Anforderungsmanagement und Change Management.