Lehrplan

REQB®
Certified Professional for Requirements Engineering

Foundation Level

Version 1.3

2011

Änderungsübersicht

Version	Datum	Kommentar
0.1	17.04.2006	Erste Version des Lehrplans, Erstellung einer Basisstruktur für den Lehrplan
0.2	20.07.2006	Erweiterung auf Version 0.1
0.3	04.09.2006	Erneute Erweiterung und Überarbeitung von Version 0.1
0.4	10.10.2006	Überarbeitete Version 0.3
0.5	15.12.2006	Überarbeitete Version 0.4
0.6	07.02.2007	Vollständig überarbeitete Version 0.5
0.7	10.04.2007	Überarbeitete Version zur Überprüfung
0.8	15.06.2007	Alpha-Version
0.9	01.09.2007	Beta-Version
1.0	15.01.2008	Freigabe von Version 1.0
1.1	29.05.2008	Aktualisierte Version 1.1
1.2	01.07.2008	Aktualisierte Version 1.2
1.3	10.07.2011	Aktualisierte Version 1.3
	30.12.2012	Deutsche Version

Grundgedanke

Das zentrale Thema dieses Lehrplans ist die zunehmende Komplexität und Abhängigkeit von Software. Daraus resultiert eine hohe Abhängigkeit von Fehlerfreiheit dieser Software. Das Requirements Engineering Qualifications Board (REQB) hat sich daher entschieden, einheitliche internationale Standards für das Requirements Engineering zu schaffen. Denn mit Standards ist es wie mit Sprachen - nur wenn man sie versteht kann man effektiv mit ihnen arbeiten. Und genau um eine solche einheitliche Sprache in diesem wichtigen Bereich des Requirements Engineering zu schaffen, sind im REQB internationale Experten zusammengekommen und haben diesen Lehrplan entwickelt.

Dank

Requirements Engineering Qualifications Board Working Party Foundation Level (Ausgabe 2011): (Karolina Zmitrowicz (chair), Alain Betro, Dorothée Blocks, Jérôme Khoualed, Eric Riou Du Cosquer, Chris Hofstetter, Michał Figarski, Francine Lafontaine, Beata Karpińska, Folke Nilsson, Ingvar Nordström, Alain Ribault, Radosław Smilgin)

Inhalt

Einführung

Zweck des Lehrplans

Dieser Lehrplan definiert die Basisstufe (Foundation Level) des Ausbildungszweigs zum REQB Certified Professional for Requirements Engineering (kurz CPRE). REQB entwickelte diesen Lehrplan in Zusammenarbeit mit der Global Association for Software Quality. Der Anwendungsbereich des REQB umfasst alle Arten von IT-bezogenen Produkten, wobei ein Produkt sowohl die Hardware und Services als auch die Software und deren Commitment-Anforderungen und Geschäftsanforderungen mit der entsprechenden Dokumentation umfasst.

Der Lehrplan dient als Grundlage für die Trainingsanbieter, die als Ausbilder akkreditiert werden möchten. Alle Abschnitte dieses Lehrplans müssen entsprechend in die Trainingsdokumente integriert sein. Der Lehrplan sollte den Teilnehmern jedoch auch als Vorbereitungsmaterial für die Zertifizierung dienen. Alle hier vorliegenden Abschnitte sind folglich auch relevant für die Prüfung, die entweder nach Abschluss eines akkreditierten Kurses oder unabhängig in einer offenen Prüfung abgelegt werden kann.

Der Lehrplan empfiehlt auch für jedes Kapitel eine Unterrichtsdauer.

Prüfung

Die Prüfung zum Certified Professional for Requirements Engineering Foundation Level basiert auf diesem Lehrplan. Alle Abschnitte dieses Lehrplans können dabei geprüft werden. Die Prüfungsfragen sind nicht notwendigerweise in die einzelnen Abschnitte unterteilt. Eine Frage kann sich auf verschiedene Abschnitte beziehen.

Die Prüfung erfolgt im Multiple Choice-Verfahren.

Prüfungen können entweder nach der Teilnahme an akkreditierten Kursen abgelegt werden oder als offene Prüfung (ohne vorherigen Kurs). Detaillierte Informationen zur Prüfungsdauer finden Sie entweder auf der Website von gasq (www.gasq.org) oder auf der REQB-Website (www.reqb.org).

Akkreditierung

Anbieter eines Kurses zum REQB Certified Professional for Requirements Engineering Foundation Level müssen von der Global Association for Software Quality akkreditiert sein. Deren Experten prüfen die Dokumentation des Anbieters auf Genauigkeit. Ein akkreditierter Kurs ist als zu diesem Lehrplan konform anerkannt. Nach Abschluss dieses Kurses kann eine offizielle Prüfung zum Certified Professional for Requirements Engineering (CPRE-Prüfung) vor einem unabhängigen Zertifizierungsinstitut (entsprechend der Regeln der ISO 17024) abgelegt werden.

Akkreditierte Trainingsanbieter sind durch das offizielle REQB-Logo für akkreditierte Trainingsanbieter zu erkennen:

Internationalität

Dieser Lehrplan wurde in einer Kooperation mehrerer internationaler Experten entwickelt. Der Inhalt dieses Lehrplans kann daher als internationaler Standard angesehen werden. Durch den Lehrplan ist es nun möglich international auf demselben Niveau zu lehren und zu prüfen.

K-Stufen

Die Lernziele dieses Lehrplans wurden in verschiedene kognitive Wissensstufen (K-Stufen) unterteilt. Dadurch ist die "Wissensstufe" zu jedem Punkt für die Kandidaten sofort ersichtlich.

Der vorliegende Lehrplan umfasst 3 K-Stufen:

- K1 - erinnern, erkennen, wiedergeben
- K2 – verstehen, erklären, Gründe angeben, vergleichen, klassifizieren, zusammenfassen
- K3 - in einem spezifischen Kontext anwenden

1 Grundlagen (K2)	40 Minuten

Lernziele für das Foundation Level des Requirements Engineering
Die Lernziele legen fest, was Sie nach Beenden des jeweiligen Moduls gelernt haben sollten.

1.1 Anforderung (K2)

LO-1.1.1 Die Definition einer Anforderung wiedergeben (K1)

LO-1.1.2 Die Bedeutung und den Zweck der Anforderungen erklären (K2)

LO-1.1.3 Erklären, wie Anforderungen klassifiziert werden können (K2)

LO-1.1.4 Die verschiedenen Arten von Anforderungen beschreiben (K2)

LO-1.1.5 Erklären, welche Probleme in Bezug auf Anforderungen bestehen (K2)

LO-1.1.6 Beschreiben, welche Konzepte in Verbindung mit Anforderungen wichtig sind (K2)

LO-1.1.7 Den Unterschied zwischen RM (Requirements Management, Anforderungsmanagement) und RE (Requirements Engineering) erklären (K2)

1.2 Standards und Normen (K1)

LO-1.2.1 Wichtige Normen und Standards für das Requirements Engineering wiedergeben (K1)

LO-1.2.2 Erklären, weshalb Requirements Engineering wichtig ist (K2)

1.1 Anforderung (K2)	20 Minuten

Begriffe:

Commitment, Kritikalität, funktionale Anforderung, nicht funktionale Anforderung, Anforderung, Requirements Engineering, Anforderungsmanagement, Anforderungsmanagement, Prozessanforderungen, Produktanforderungen, Priorität, Lösung, Validierung, Verifizierung

1.1.1 Definition und Klassifizierung (K2)

Definition der Bedeutung des Begriffs „Anforderung“ (K1):

Anforderung [IEEE 610.12]:

- Eine Bedingung oder Fähigkeit, die ein Benutzer benötigt, um ein Problem zu lösen oder ein Ziel zu erreichen.
- Eine Bedingung oder Fähigkeit, die ein System oder eine Systemkomponente erfüllen oder besitzen muss, um einen Vertrag, einen Standard, eine Spezifikation oder anderweitig formal bereitgestellte Dokumente einzuhalten.
- Eine dokumentierte Darstellung einer Bedingung oder Fähigkeit wie unter (1) oder (2).

Bedeutung und Zweck von Anforderungen (K2):

- Grundlage für die Bewertung, Planung, Ausführung und Überwachung der Projektaktivität
- Kundenerwartungen
- Komponente der Vereinbarungen, Aufträge, Projektpläne etc.
- Festlegen von Systemgrenzen, Lieferumfang, vertraglich vereinbarter Services

Klassifizierung von Anforderungen [Ebert05] (K2)

Anforderungen bestehen aus:

- Prozessanforderungen
- Produktanforderungen

Prozessanforderungen sind Anforderungen, die sich auf die Entwicklung und die Lieferprozesse beziehen. Beispiele: Kosten, Marketing, Verarbeitungszeit, Vertrieb und Distribution, Organisation, Dokumentation.

Produktanforderungen setzen sich aus funktionalen und nicht funktionalen Produktanforderungen zusammen. Beide können aus Sicht des Benutzers (extern) oder Kunden und aus Sicht des Entwicklungsteams (intern) betrachtet werden. Es ist wichtig, zu wissen, dass Benutzer und Kunden verschieden sein können.

Funktionale Anforderungen beschreiben die Funktion (das Verhalten) des Systems.

Nicht funktionale Anforderungen beschreiben die Qualitätsattribute des Systems. Sie werden oft als „Qualitätsattribute“ bezeichnet.

Der Unterschied zwischen funktionalen und nicht funktionalen Anforderungen kann durch die folgenden Aussagen ausgedrückt werden:

- Funktionale Anforderungen beschreiben, *was* das System tut.
- Nicht funktionale Anforderungen beschreiben, *wie* das System etwas tut.

Beispiele:

Funktionale Produktanforderungen aus Sicht des Benutzers und Kunden: Benutzeroberfläche, Anwendungen, Services.

Funktionale Produktanforderungen aus Sicht des Entwicklungsteams: Architektur, Stromversorgung, Lastverteilung.

Nicht funktionale Produktanforderungen aus Sicht des Benutzers und Kunden: Zuverlässigkeit, Leistung, Benutzerfreundlichkeit

Nicht funktionale Produktanforderungen aus Sicht des Entwicklungsteams: Testbarkeit, Gebrauchstauglichkeit, Werkzeuge

<u>Grundlegende Arten von Anforderungen (K1):</u>

- Kundenanforderungen
    - Wünsche, Bedürfnisse und Erwartungen des Kunden (abstrakte Geschäftsanforderungen)
    - Einschränkungen für das Geschäft
- Lösungs-/Systemanforderungen
    - Angabe der Kundenbedürfnisse (detaillierte Angabe der abstrakten Geschäftsanforderungen)
- Produkt-/Komponentenanforderungen

- Funktionen und Eigenschaften der Lösung
- Grundlage für detaillierte Analyse und Ausführung (Beispiel: Systemanwendungsfälle)

1.1.2 Probleme mit Anforderungen (K1)

Häufigste Probleme mit Anforderungen:

- Unklare Ziele
- Kommunikationsprobleme
- Sprachbarrieren
- Wissensgrenzen
- Vage Formulierungen
- Zu formale Formulierungen
- Unbeständigkeit der Anforderungen
- Schlechte Qualität der Anforderungen (z. B. zweideutige, übermäßig spezifizierte, unklare, unmögliche, widersprüchliche Anforderungen)
- Ausschmückungen (zu viel und unnötige Beschreibungen zu einer Anforderung, wodurch die eigentliche Anforderung nicht mehr klar zu erkennen ist)
- Unzureichende Beteiligung der Benutzer
- Übersehene Benutzerklassen (resultierend aus fehlenden Stakeholdern)
- Ungenaue Planung
- Minimale Spezifikation

1.1.3 Qualitätskriterien für Anforderungen (K2)

Qualitätskriterien für Anforderungen [Wiegers05] (K2):

1. Alle Anforderungen müssen die folgenden Eigenschaften aufweisen:

- Korrekt – die Anforderung muss die erforderliche Funktionalität akkurat beschreiben. Der Referenzpunkt zur Evaluierung der Korrektheit ist die Quelle der Anforderung (zum Beispiel Kunden oder eine abstrakte Systemanforderung).
- Realisierbar – die Anforderung muss innerhalb der bekannten Kapazitäten und/oder Beschränkungen des Systems und der Umgebung implementiert werden können.

- Notwendig – die Anforderung sollte dokumentieren, was der Kunde (oder ein anderer Stakeholder) wirklich benötigt und was zur Erfüllung einer externen Anforderung oder Schnittstelle oder eines angegebenen Standards erforderlich ist.
- Priorisiert – der Anforderung sollte eine Priorität zugewiesen werden, die angibt, wie wichtig sie für eine bestimmte Produktfreigabe ist.
- Unzweideutig – die Anforderung sollte nur auf eine Weise interpretiert werden können. Verschiedene Leser einer Anforderung sollten die Anforderung auf gleiche Weise interpretieren und verstehen.
- Überprüfbar – es sollte möglich sein, zu überprüfen, ob die Anforderung korrekt implementiert ist.
- Einzeln – enthält nicht mehrere Anforderungen; impliziert genügend Granularität zur Angabe einer einzelnen Anforderung.
- Unabhängig von der Ausführung – beschreibt „Was“ und nicht „Wie“

2. Die Anforderungsspezifikation muss folgendermaßen erstellt sein:

- Vollständig – keine Anforderungen oder nötigen Informationen sollten in der Anforderungsspezifikation fehlen. Vollständigkeit ist auch als gewünschte Eigenschaft einer einzelnen Anforderung und Detaillierungsstufe ausgedrückt.
- Konsistent – die Anforderung darf keinen Konflikt mit anderen Softwareanforderungen oder mit abstrakten (System- oder Geschäfts-)Anforderungen verursachen.
- Modifizierbar – die Spezifikation muss Änderungen an den Anforderungen zulassen. Für jede Anforderung sollte ein Änderungsverlauf gepflegt werden.
- Nachvollziehbar – für jede Anforderung sollte der Bezug zu ihrer Quelle (beispielsweise abstrakte Systemanforderung, Anwendungsfall oder Kundenaussage) und die darauf bezogenen Implementierungsartefakte (beispielsweise Ausführungselemente, Quellcode und Testfälle) herstellbar sein.

1.1.4 Lösung (K1)

Lösung (K1)

Eine Lösung ist die Implementierung der Anforderung.

Eine Lösung kann ein Softwaresystem, eine Prozessverbesserung oder dergleichen sein.

1.1.5 Commitment (K1)

Commitment (K1)

Commitment bezeichnet den Grad des Engagements zur Erfüllung der Anforderung.

Das Commitment ist definiert über Schlüsselwörter, die mit höheren Anforderungen verknüpft sind:

- Das System sollte…

Mögliche Schlüsselwörter: „muss", „wird", „sollte", „würde", „könnte".

Die Schlüsselwörter „muss", „sollte", „würde", „könnte" beziehen sich auf Geschäfts- und Benutzeranforderungen vor der Vereinbarung. Nach der Vereinbarung und der Festlegung der Anforderungen sollten die Schlüsselwörter den Grad des Engagements genauer bestimmen:

- Das System wird…

Der Grad des Engagements für eine Anforderung kann anhand der MoSCoW-Notation (Must have, Should have, Could have, Would have) ausgedrückt werden.

Wenn der Lösungsanbieter und der Kunde eine Vereinbarung treffen, wird das Commitment für die Anforderungen von den Projektteilnehmern abgerufen.

Anforderungen können sich im Lauf eines Projekts weiterentwickeln. Wenn sich die Anforderungen weiterentwickeln, wird durch Abrufen des Commitments für Anforderungen von den Projektteilnehmern sichergestellt, dass sich die Projektteilnehmer für die aktuellen und vereinbarten Anforderungen und die daraus resultierenden Änderungen an den Projektplänen, Aktivitäten und Arbeitsprodukten engagieren.

1.1.6 Rechtliche Zuständigkeiten und Fehler (K1)

Rechtliche Zuständigkeiten (K1)

Es gibt rechtliche Zuständigkeiten, die sich auf die Qualität der Software beziehen. Rechtliche Zuständigkeiten beziehen sich oft auf eine bestimmte Anforderung (z. B. Umweltanforderung für ein Kernkraftwerk, ein Flugzeug), die vom gelieferten Produkt erfüllt werden müssen. Die Zuständigkeiten können sich auch auf Fehler im Produkt beziehen.

Rechtliche Zuständigkeiten sollten im Vertrag zwischen dem Verkäufer und dem Kunden definiert werden. In einigen Branchen kann es auch erforderlich sein, vertragliche oder gesetzliche Vorgaben oder spezielle Industrienormen zu erfüllen.

Fehler (K1)

Ein Fehler ist ein Makel in einer Komponente oder einem System, aufgrund dessen die Komponente oder das System die erforderliche Funktion nicht ausführen kann, wie beispielsweise eine falsche Aussage oder Datendefinition.

Ein Fehler, der bei der Ausführung auftritt, kann zum Ausfall der Komponente oder des Systems führen [ISTQB].

1.1.7 Priorität und Kritikalität von Anforderungen (K1)

Priorität von Anforderungen (K1)

Priorität ist eine Evaluierung der Wichtigkeit/Dringlichkeit einer Anforderung.

Laut [SWEBOK] gilt im Allgemeinen: Je höher die Priorität, desto entscheidender ist die Anforderung zur Erfüllung der Gesamtziele der Software. Die Priorität wird oft auf einer festen Skala klassifiziert (obligatorisch, sehr erwünscht, erwünscht oder optional) und muss oft mit den Entwicklungs- und Implementierungskosten ausgeglichen werden.

Beispiele für Prioritäten bei Anforderungen:

- Hoch
- Mittel
- Niedrig

Kritikalität von Anforderungen (K1)

Evaluierung des Risikos einer Anforderung durch Evaluieren des Schadens im Fall der Nichterfüllung einer Anforderung.

Die Kritikalität wird in Stufen ausgedrückt; je höher die Stufe, desto schwerwiegender die Konsequenzen im Fall eines funktionalen Fehlers.

1.1.8 Validierung und Verifizierung (K1)

Die Validierung ist ein Vorgang der Bestätigung, dass die Spezifikation einer Phase oder das gesamte System die Kundenanforderungen erfüllt.

Die Validierung wird normalerweise mit Unterstützung des Kunden durchgeführt und soll bestätigen, dass die Anforderungen oder die Anforderungsspezifikation beschreibt, was der Kunde benötigt.

Laut CMMI demonstrieren die Validierungsaktivitäten, dass ein Produkt oder eine Produktkomponente die beabsichtigte Funktion erfüllt, wenn sie in der vorgesehenen Umgebung eingesetzt wird. Dadurch wissen Sie, dass Sie “das Richtige getan haben". Kunden können Produktbeschreibungen oder Anforderungen nur annähernd verstehen und die Validierung hilft dabei, zu erkennen, was benötigt wird (durch den Einsatz von Werkzeugen wie Szenarien, Anwendungsfälle, Prototyperstellung etc.). Die Verifizierung ist ein Vergleich eines Zwischenprodukts mit dessen Spezifikationen. Dabei wird bestimmt, ob die Software korrekt

entwickelt wurde und ob die Spezifikationen, die in der vorigen Phase festgelegt wurden, erfüllt wurden.

Laut CMMI bietet die Verifizierung Kontrollpunkte, an denen ausgewählte Arbeitsergebnisse oder Zwischenprodukte überprüft werden, um zu bestätigen, dass sie die Anforderungen erfüllen. Verifizierungsaktivitäten konzentrieren sich auf die schrittweise Bestätigung der Implementierung von Anforderungen; sie ermöglichen die frühzeitige und fortlaufende Bestätigung, dass das Produkt richtig entwickelt wird.

Die üblichsten Methoden zur Verifizierung und Validierung sind Reviews, Audits, Checklisten und Tests.

Der Unterschied zwischen Validierung und Verifizierung kann wie folgt ausgedrückt werden:

- Verifizierung – haben wir das Produkt richtig erstellt?
- Validierung – haben wir das richtige Produkt erstellt?

1.1.9 Requirements Engineering, Anforderungsmanagement und Anforderungsentwicklung (K2)

Abgrenzung zwischen Anforderungsmanagement, Anforderungsentwicklung und Requirements Engineering (K2)

Requirements Engineering (RE) ist eine Unterdisziplin des Software Engineerings und konzentriert sich auf die Festlegung und die Verwaltung von Anforderungen von Hardware- und Softwaresystemen. Das Requirements Engineering umfasst das Anforderungsmanagement und die Anforderungsentwicklung.

Zur Disziplin des Requirements Engineering gehören die folgenden Unterprozesse: Anforderungserhebung, -analyse und -verhandlung (einschließlich Anforderungspriorisierung), Spezifikation, Systemmodellierung, Anforderungsvalidierung.

Diese Unterprozesse können sich überlappen. So kann beispielsweise die Systemmodellierung sowohl ein Teil der Analyse als auch der Spezifikation sein, in einigen Fällen sogar der Erhebung.

Das Anforderungsmanagement stellt einen Arbeitsrahmen für das Requirements Engineering dar und ist über Schnittstellen mit anderen Prozessen verbunden, wie dem Projektmanagement, dem Konfigurationsmanagement und dem Qualitätsmanagement.

Der Zweck des Anforderungsmanagement besteht darin, die Anforderungen der Projektprodukte und Komponenten zu verwalten, die Abstimmung zwischen diesen Anforderungen, den Projektplänen und den Arbeitsprodukten im gesamten Produktlebenszyklus (Entwicklungszyklus und Wartungszyklus) eines Projekts sicherzustellen.

Das Anforderungsmanagement (Requirements Management, RM) umfasst Prozesse für das Management der Anforderungen insgesamt. Es ist ein ständiger Prozess der Dokumentation, Analyse, Rückverfolgung, Priorisierung, Kommunikation, Vereinbarung von Anforderungen und der Steuerung von Anforderungsänderungen.

Die Anforderungsentwicklung stellt eine Sammlung von Aktivitäten, Aufgaben, Methoden und Werkzeugen zur Erkennung, Analyse und Validierung von Anforderungen dar. Dazu gehört auch der Prozess der Transformierung von Bedürfnissen in Anforderungen.

Der Zweck der Anforderungsentwicklung besteht darin, Kunden-, Produkt- und Produktkomponentenanforderungen zu erheben, zu analysieren, zu erstellen und zu validieren.

1.2 Standards und Normen (K1)	20 Minuten

Um die Prüfung zu bestehen, ist es nicht erforderlich, den Inhalt aller Normen zu kennen. Es ist jedoch wichtig zu wissen (K1), welche Normen für das Requirements Engineering wichtig sind.

1.2.1 Standards (K1)

ISO 9000:

Anforderungen eines Qualitätsmanagementsystems:

- Definierte Konzepte und Grundlagen eines QMS
- Domänen- oder branchenneutral

ISO 9126 (ersetzt durch ISO/IEC 25000):

Definiert ein Qualitätsmodell mit sechs Kategorien: Funktionalität, Zuverlässigkeit, Gebrauchstauglichkeit, Effizienz, Wartungsfreundlichkeit, Übertragbarkeit

IEEE 610:

Standardglossar der Software Engineering-Terminologie

IEEE 830:

Empfohlene Praxis für Softwareanforderungsspezifikationen

IEEE 1233:

Richtlinie zur Entwicklung von Systemanforderungsspezifikationen

IEEE 1362:

Richtlinie für Informationstechnologie – Systemdefinition

SWEBOK - The Guide to the Software Engineering Body of Knowledge (bekannt als technischer ISO-Report 19759):

SWEBOK beschreibt das allgemein anerkannte Wissen über Software Engineering. In den 10 Wissensbereichen sind grundlegende Konzepte zusammengefasst einschließlich einer Referenzliste mit Hinweisen zu detaillierten Informationen.

1.2.2 Prozessnormen (K1)

ISO 12207:

Standard für den Softwarelebenszyklusprozess

ISO 15288:

Systemlebenszyklusprozess

ISO 15504:

Software Process Improvement and Capability Determination (SPICE)

1.2.3 Die Gründe für eine Vernachlässigung des Requirement Engineering (K2)

Das Requirements Engineering ist enorm wichtig. Und dennoch wird es immer wieder vernachlässigt.

Mögliche Gründe für die Vernachlässigung des Requirements Engineering (K2):

- Hoher Zeitdruck
- Ausschließliche Ausrichtung auf schnelle Ergebnisse
- Ausschließliche Fixierung auf die Kosten
- Ausschließliche Beachtung von funktionalen Anforderungen
- Fehlinterpretationen (viele Dinge werden als gegeben angenommen) und fehlendes Verständnis der Wichtigkeit des Requirements Engineering für den Erfolg eines Projekts

Mögliche Folgen der Vernachlässigung des Requirement Engineering (K2):

- Anforderungen werden unpräzise
- Anforderungen sind zweideutig

- Anforderungen sind widersprüchlich
- Anforderungen, die sich bei der Softwareentwicklung oft ändern
- Anforderungen, die nicht den Kriterien entsprechen
- Anforderungen, die unterschiedlich interpretiert werden können
- Fehlende Anforderungen

2 Prozessmodelle und Requirements Engineering-Prozess (K2)	60 Minuten

Lernziele für das Foundation Level des Requirements Engineering
Die Lernziele legen fest, was Sie nach Beenden des jeweiligen Moduls gelernt haben sollten.

2.1 Prozessmodelle (K2)

LO-2.1.1 Die verschiedenen Prozessmodelle beschreiben (K2)

LO-2.1.2 Erklären, wie sich die verschiedenen Prozessmodelle unterscheiden (K2)

2.2 Requirements Engineering-Prozess (K2)

LO-2.2.1 Die Eigenschaften des Requirements Engineering-Prozesses beschreiben (K2)

LO-2.2.2 Die Phasen des Requirements Engineering-Prozesses beschreiben (K2)

2.1 Prozessmodelle (K2)	30 Minuten

Begriffe:

Extremprogrammierung (Extrem Programming), Prozessmodelle, Produktlebenszyklus, Rational Unified Process, V-Modell

2.1.1 Prozessmodelle (K2)

Prozessmodelle sind von Methoden unabhängige Prozessbeschreibungen von Entwicklungsprozessen.

Rollen, Aktivitäten, Phasen und Dokumente werden dabei berücksichtigt.

Ein Softwareprozessmodell gibt das Standardformat für die Planung, Strukturierung und Ausführung eines Softwareprozesses vor.

Produktlebenszyklus (Product Life Cycle, PLC) (K2)

Definiert verschiedene Phasen der Produktentwicklung.

Grundlegende Phasen:

1. Planung
2. Entwicklung
3. Pflege
4. Lebensende

Die Planungsphase umfasst: Vision, Strategie, Business-Plan und Kosten/Nutzen-Analyse

Die Entwicklungsphase umfasst: Spezifikation, Entwurf und Implementierung. Die Entwicklungsphase ist oft in die folgenden vier Phasen unterteilt:

- Analyse
- Design
- Implementierung
- Test

2.1.2 Allgemeines V-Modell (K2)

Entwicklungsschritte:

- Definition der Anforderung, Festlegung der Anforderung (abstrakte Anforderungsspezifikationen)
- Funktionaler Systementwurf, Systemanalysen (funktionale Spezifikationen)
- Technischer Systementwurf, Architekturentwurf (Softwaredesign)
- Komponentenspezifikation
- Implementierung

Dieses Modell ist V-förmig aufgebaut und mit jeder Stufe ist eine Teststufe verknüpft:

Anforderungsdefinition und -analyse	→	Abnahmetest
Funktionales Systemdesign	→	Systemtest
Technische Ausführung	→	Integrationstest
Komponenten- (Modul-) entwurf	→	Unit-Test

Implementierung

Für Schulungsunternehmen: Grafisches Porträt des allgemeinen V-Modells; detaillierte Beschreibung des allgemeinen V-Modells

2.1.3 Rational Unified Process (RUP©) (K2)

Rational Unified Process ist die Bezeichnung für ein Prozessmodell von IBM Rational ©

Es ist ein iteratives Modell (d.h. der Prozess wird so lange wiederholt bis alle Szenarien, Risiken und Änderungen behandelt wurden). Es umfasst 9 Disziplinen einschließlich eine Anforderungsdisziplin (6 Engineering-Disziplinen plus 3 unterstützende Disziplinen). Jede Disziplin wird von 4 aufeinanderfolgenden Phasen des Projektlebenszyklus abgedeckt: Inception (erste Konzeptionsphase), Elaboration (Planung der Konstruktionsphase), Construction (Produktentwicklung und –test), Transition (Übergabe und Auslieferung der Software).

Für Schulungsunternehmen: Vertiefung des RUP© mit grafischer Präsentation; vertieftes Studium der Anforderungsdisziplin.

2.1.4 Agile Ansätze

Anforderungsmanagement

In agilen Umgebungen werden Anforderungen über Mechanismen wie „Product Backlogs“, „Story Cards“ und „Screen Mock-ups“ kommuniziert und verfolgt. Commitments für Anforderungen werden entweder kollektiv durch das Team vorgenommen oder durch einen damit beauftragten Teamleiter. Arbeitszuweisungen werden regelmäßig (z. B. täglich, wöchentlich) basierend auf den Fortschritten und als verbessertes Verständnis der Anforderungen und Lösungsentwicklung angepasst. Rückverfolgbarkeit und Konsistenz in den Anforderungen und Arbeitsprodukten wird über die bereits genannten Mechanismen behandelt sowie während der Aktivitäten beim Iterationsstart oder -ende wie „Retrospektiven“ und „Demo-Days“.

Anforderungsentwicklung

In agilen Umgebungen werden Kundenbedürfnisse und -ideen iterativ erfasst, ausgearbeitet, analysiert und validiert. Anforderungen werden in Form von User Storys, Szenarien, Anwendungsfällen, Product Backlogs und den Ergebnissen von Iterationen (Arbeitscode im Fall von Software) dokumentiert. Welche Anforderungen in einer vorliegenden Iteration behandelt werden, hängt von einer Risikobewertung ab und von den Prioritäten, die mit dem verknüpft sind, was noch im Product Backlog zu verarbeiten ist. Welche Details der Anforderungen (und anderer Artefakte) dokumentiert werden sollen, hängt von der Notwendigkeit der Koordination (zwischen den Teammitgliedern, Teams und späteren Iterationen) ab und vom Risiko des Verlusts dessen, was bereits gelernt wurde. Wenn der Kunde im Team mitarbeitet, kann es noch erforderlich sein, die Kunden- und Produktdokumentation zu trennen, um die Erforschung mehrerer Lösungen zu ermöglichen. Bei der weiteren Lösungsentwicklung werden die Zuständigkeiten für abgeleitete Anforderungen den entsprechenden Teams zugewiesen.

2.1.5 Extreme Programming (Extremprogrammierung) (K2)

„Extreme Programming” oder Extremprogrammierung ist eine von Kent Beck und anderen entwickelte Methode zur Softwareentwicklung, die die Zuständigkeiten für Änderungen an Kundenanforderungen regelt. Das Projektmanagement (z. B. Scrum, Abschnitt 2.1.6.) definiert wie und wann die Änderungen an Kundenanforderungen in der Softwarelösung implementiert werden (z. B. in welchem Sprint). Es empfiehlt dem Kunden häufige Software-Releases innerhalb kurzer Entwicklungszyklen (Timeboxing), was die Produktivität verbessern und Kontrollpunkte, an denen neue Anforderungen übernommen werden können, ermöglichen soll.

Einige Eigenschaften der Extremprogrammierung:

- Paarprogrammierung
- Umfassende Code-Überprüfungen
- Einheitentests des Codes

- Vermeidung der Programmierung von Funktionen, die nicht unbedingt gebraucht werden
- Flache Verwaltungsstruktur (keine komplexe Hierarchie im Team). Dies wird am besten in Scrum-Teams erreicht – das Scrum-Team ist „selbstverwaltet“, es gibt keine Rollen wie Leiter, PMs etc.)
- Einfachheit und Klarheit im Code
- Erwarten von Änderungen an den Kundenanforderungen im Lauf der Zeit und besseres Verständnis des Problems
- Häufige Kommunikation mit dem Kunden und den Programmierern untereinander
- Vollständiger Verzicht auf Festlegung der Anforderung (keine separate „Anforderungsphase” vor dem Entwicklungsstart; Anforderungsentwicklung, -verfeinerung und -erkennung sind Teil der eigentlichen Softwareentwicklung (Programmierung))

2.1.6 Scrum (K2)

Scrum ist die Bezeichnung für ein agiles Vorgehensmodell. Scrum wurde ursprünglich für Softwareentwicklungsprojekte herausgebracht. Scrum enthält Praxisanleitungen und vordefinierte Rollen. Die Hauptrollen in Scrum:

- Scrum Master (verantwortlich für die Pflege des Prozesses)
- Product Owner (repräsentiert die Stakeholder und das Unternehmen)
- Team (eine funktionsübergreifende Gruppe, die die eigentliche Analyse, Entwicklung, Implementierung und Testaufgabe etc. ausführt)

Eine der Hauptaufgaben von Scrum ist die Unterteilung der Entwicklung in sogenannte „Sprints“ (normalerweise jeweils ein Zeitraum von zwei bis vier Wochen). Bei jedem Sprint erstellt das Team sogenannte potenzielle lieferbare Produktschritte.

Scrum ermöglicht die Verwaltung von Anforderungen über „Backlogs“. Es gibt zwei Arten von Backlogs:

- Product Backlog – eine detaillierte Liste, die während des gesamten Projekts gepflegt wird. In ihr werden Anforderungen in Form von ausführlichen Beschreibungen aller potenziellen Funktionen priorisiert nach Geschäftswert gesammelt. Der Product Backlog ist Eigentum des Product Owners.
- Sprint Backlog – die Liste der Arbeit, die im nächsten Sprint vom Team erledigt werden soll. Funktionen werden in Aufgaben unterteilt, die im Idealfall normalerweise vier bis sechzehn Stunden Arbeit ausmachen sollen. Der Sprint Backlog ist Eigentum des Teams.

Haupteigenschaften des Scrum-Ansatzes:

- Am Ende jedes Sprints sollte eine funktionale Software geliefert werden – in der Praxis beginnen die Teams mit der Arbeit an der Anforderungsanalyse und fahren während des eigentlichen Sprints damit fort. Bei der Aufarbeitung der Aufgaben klären sich die Anforderungen.
- Von den Teammitgliedern wird Zusammenarbeit erwartet und die regelmäßige Einbeziehung des Kunden ist ein Schlüsselkonzept der agilen Entwicklung. Dem Kunden kann in Demo-Meetings zur neuen Softwareversion gegen Ende eines Sprints ein Feedback gegeben werden; dies kann aber auch über Benutzerabnahmetests erfolgen.
- Anforderungen ergeben sich aus dem Kundenfeedback – wenn Kunden die Software in der Ausführung sehen, können sie ihre Anforderungen klären.
- Anforderungen werden vor dem Entwicklungsstart nicht vollständig spezifiziert, doch die für einen Sprint ausgewählten Funktionen werden zu Beginn des Sprints angegeben.
- Erwartungsgemäß werden die Funktionen im Verlauf des Projekts ständig neu priorisiert.

Der wesentliche Einfluss von Scrum auf das Requirements Engineering besteht darin, dass die Anforderungsspezifikationen nicht vor Beginn der Projektentwicklung abgeschlossen und validiert werden.

Der Product Owner und das Team wählen gemeinsam diejenigen Funktionen aus dem Product Backlog aus, die im nächsten Sprint auf Basis der Geschäftspriorität und des erforderlichen Arbeitsaufwands bearbeitet werden sollen. Benutzeranforderungen werden vom Product Owner als „User Storys" formuliert, die Informationen über das „Wer, Was, Warum", nicht jedoch das „Wie" einer Anforderung enthalten. Zu Beginn eines Sprints werden die ausgewählten Funktionen in die Aufgaben des Sprint Backlogs unterteilt und anschließend entwickelt.

Durch Einbeziehen der Product Owners, zum Beispiel durch Präsentation der implementierten Funktionen der Software, können diese die Anforderungen für das Team und sich selbst klären.

Für Schulungsunternehmen: Nennen Sie Beispiele für User Storys und die entsprechenden Elemente der Product und Sprint Backlogs.

Für Schulungsunternehmen: Erklären Sie auch mindestens zwei weitere agile Modelle einschließlich „Crystal".

2.1.7 Reifegradmodell (K2)

Reifegrade dienen zur Erkennung und Verbesserung der Prozessreife (Prozessbeurteilung und Prozessverbesserung).

ISO/IEC 15504 (SPICE – Software Process Improvement and Capability Determination) (K1)

ISO/IEC 15504 (SPICE – Software Process Improvement and Capability Determination bezeichnet eine Reihe von technischen Standards für den Softwareentwicklungsprozess und die darauf bezogenen Funktionen für das Geschäftsmanagement.

ISO/IEC 15504 kann als Methode zur Prozessverbesserung und/oder Ermittlung von Fähigkeiten (zum Beispiel Bewertung der Prozessfähigkeit des Anbieters) eingesetzt werden.

SPICE definiert Prozesse, die in fünf Prozesskategorien unterteilt sind:

- Kundenlieferant
- Engineering
- Unterstützung
- Management
- Organisation

Für die oben genannten Prozesse definiert die ISO/IEC 15504 jeweils eine Fähigkeitsstufe:

0. Unvollständiger Prozess
1. Durchgeführter Prozess
2. Gesteuerter Prozess
3. Etablierter Prozess
4. Vorhersagbarer Prozess
5. Optimierter Prozess

Capability Maturity Model Integrated (CMMI)

Definiert fünf Reifegrade für Entwicklung, Services und Beschaffung.

1. Initial (beginnend: chaotisch, ad hoc, individuelle Anstrengungen) – der Grundzustand der Verwendung eines neuen Prozesses.
2. Managed (gesteuert) – der Prozess wird gemäß vereinbarter Metriken gesteuert.
3. Defined (definiert) – der Prozess wird als Standardgeschäftsprozess definiert/bestätigt und in die Stufen 0, 1 und 2 unterteilt.
4. Quantitatively managed (quantitativ gesteuert)
5. Optimizing (optimierend) – das Prozessmanagement schließt eine bewusst gewollte Prozessoptimierung/-verbesserung mit ein.

Für Schulungsunternehmen: Vertiefung von ISO 15504/SPICE und CMMI; mit einer Beschreibung der typischen Anforderungen für das Requirements Engineering

2.2 Requirements Engineering-Prozess (K2)	30 Minuten

Begriffe:

Kundenorientierter Prozess, Perspektive, Requirements Engineering

2.2.1 Definition des Requirements Engineering-Prozesses (K2)

Requirements Engineering ist eine Disziplin mit Prozessen, die zur Erkennung, Strukturierung und Steuerung von Anforderungen erforderlich sind. Zum Requirements Engineering gehören die folgenden Unterprozesse:

- Identifizierung der Anforderungen
- Analyse der Anforderungen
- Spezifikation der Anforderungen
- Vereinbarung der Anforderungen
- Änderungen an den Anforderungen
- Validierung und Qualitätssicherung

2.2.2 Einflüsse auf das Requirements Engineering

Einige Faktoren können einen negativen Einfluss auf das Requirements Engineering ausüben:

- Intern (in der Organisation des Softwareanbieters):
  - Keine Kenntnis der Domain des Benutzers
  - Ineffektiver Ansatz bzw. ineffektive Methode des Requirements Engineering
  - Unzureichende Erfahrung und Kenntnisse der Mitarbeiter
- Extern (außerhalb der Organisation des Softwareanbieters):
  - Fehlende Kommunikation
  - Unklare und/oder wechselnde Geschäftsziele und dadurch instabile Anforderungen
  - Keine Kenntnisse über den Softwareentwicklungsprozess
  - Kein Einbeziehen der Benutzer und/oder Stakeholder

Es gibt verschiedene Stakeholder mit unterschiedlichen Ansichten über den Requirements Engineering-Prozess. Im Allgemeinen kann man den Prozess aus Sicht des Kunden und aus Sicht des Lieferanten (Anbieters) sehen.

Beispiel:

Aus Sicht des Kunden können die wichtigsten Aspekte im Requirements Engineering die Benutzeroberfläche, Anwendungen und Services sein. Aus Sicht des Anbieters sind andere wichtige Aspekte die Architektur, die Lastverteilung etc.

Methode des Requirements Engineering-Prozesses mit dem Kunden im Mittelpunkt:

- Kundenorientierte Analyse und Design
- Prototyping-Ansatz
- Demos als Mittel zu Validierung der Teilschritte des Systems

3 Projekt- und Risikomanagement (K2)	60 Minuten

Lernziele für das Foundation Level des Requirements Engineering
Die Lernziele legen fest, was Sie nach Beenden des jeweiligen Moduls gelernt haben sollten.

3.1 Projektmanagement (K2)

LO-3.1.1 Erklären, weshalb das Requirements Engineering für Projekte wichtig ist (K2)

LO-3.1.2 Die Fehler wiedergeben, die im Requirements Engineering auftreten können (K1)

3.2 Risikomanagement (K3)

LO-3.2.1 Die Risiken in Bezug auf das Requirements Engineering erkennen (K1)

3.1 Projektmanagement (K2)	30 Minuten

Begriffe:

Projektkonzeption, Vertragsverhandlungen, Projektdefinition, Projektausführung

3.1.1 Notwendigkeit des Requirements Engineering in Projekten (K2)

Einige der Gründe, weshalb Projekte nicht gelingen, haben mit den Anforderungen zu tun. Eine Vernachlässigung des Requirements Engineering kann dazu führen, dass Anforderungen nicht präzise sind oder widersprüchlich, oder sie erfüllen nicht die Kriterien und entsprechen nicht den Bedürfnissen der Stakeholder. Daher ist ein sorgfältiges und strukturiertes Requirements Engineering ein notwendiger Teil eines jeden Projekts.

Das Requirements Engineering sollte einen zu den folgenden Bereichen leisten (K1):

- Projektkonzeption
  - Erkennen der Bedürfnisse und Erwartungen der Kunden hinsichtlich der Problemlösung
  - Erstellen von abstrakten Anforderungen
- Vertragsverhandlungen
  - Evaluierung von Kundenanforderungen
  - Bestimmen des ursprünglichen Umfangs und der erforderlichen Ressourcen für das Projekt
  - Ermitteln der Entwicklungskosten (Kosten für die Implementierung der Anforderungen)
  - Vereinbaren der Prioritäten für die Anforderungen
- Projektdefinition
  - Definition von Rollen, Aufgaben, Aktivitäten und zusätzlichen Prozessen (Beispiel: Änderungsmanagement)
  - Detaillierte Geschäftsausführung der Lösung
  - Beitrag zum Architekturdesign
  - Beitrag zum Testen von Prozessergebnissen

- Projektausführung
    - Schaffen einer Basis für die Anforderungsentwicklung und Anforderungsüberprüfung und -validierung (Tests)
    - Erzwingen von Review-Plänen und deren Anpassung an den aktuellen Umfang der Lösung im Fall von Änderungen an den Anforderungen

3.1.2 Welche Fehler können beim Requirements Engineering auftreten? (K2)

Häufigste Fehler:

- Unklare Anforderungen
- Änderungen an den Anforderungen (Änderungen aufgrund von unklaren Projektzielen und der fehlenden Kenntnis der Geschäftsdomäne des Kunden; Anforderungsänderungen werden im agilen und iterativen Ansatz nicht als Fehler wahrgenommen)
- Instabile Produkt- und Designbasis für Teilaufträge
- Unklare Verantwortlichkeiten (sowohl beim Kunden als auch beim Anbieter)
- Keine Übereinstimmung der Erwartungen des Kunden mit den Projektinhalten
- Ungenügendes Einbeziehen des Kunden
- Projektdefinition mit Meilensteinen, die nicht erreicht werden können
- Ungenaue Ausgabeneinschätzung
- Ungenaue Abschätzung der Auswirkungen der Anforderungsänderungen auf die anderen Bereiche des Produkts in der Entwicklung
- Keine Rückverfolgbarkeit

3.2 Risikomanagement (K2)	30 Minuten

Begriffe:

Failure Mode and Effect Analysis (Fehlermöglichkeits- und Einflussanalyse, FMEA), Produktrisiko, Risiko, Risikomanagement, Risikomanagementplan

3.2.1 Notwendigkeit des Risikomanagements (K2)

Effizientes Risikomanagement als Schlüssel zur Minimierung der Projekt- und Produktrisiken. Die Erkennung und sorgfältige Analyse von Risiken und die Planung der angemessenen Reaktion darauf minimiert die Möglichkeit, dass ein Risiko auftritt sowie die Konsequenzen in diesem Fall.

3.2.2 Risiko (K2)

Risiko (K1)

Ein Risiko wird als Auswirkung aus ungewissen Zielen definiert, sei es positiv oder negativ [ISO 31000].

Eine andere Definition beschreibt ein Risiko als die Möglichkeit eines Ereignisses, einer Gefahr oder Bedrohung oder einer Situation, die auftreten kann und unerwünschte Konsequenzen hat oder ein potenzielles Problem verursacht. Die Risikostufe wird durch die Wahrscheinlichkeit eines nachteiligen Ereignisses und dessen Auswirkungen (der Schaden durch dieses Ereignis) [ISTQB] bestimmt.

Risikotypen (K2)

Es gibt zwei Arten von Risiken:

- Produktrisiko
- Projektrisiko

Projektrisiken (K2)

Projektrisiken sind Risiken, die damit zusammenhängen, ob die Projektziele erreicht werden können. Beispiele:

- Organisatorische Faktoren:
  - Ausbildung, Schulung und Personalmangel

    - Probleme mit den Mitarbeitern
    - Politische Probleme wie zum Beispiel:
        - Probleme der Stakeholder bei der Kommunikation ihrer Bedürfnisse und Erwartungen
        - Keine Aufarbeitung der Information aus Reviews (Beispiel: Keine Verbesserung der Praktiken zur Dokumentation von Anforderungen)
    - Falsche Haltung oder Erwartungen an das Requirements Engineering
- Technische Probleme:
    - Probleme bei der Definition der richtigen Anforderungen
    - Das Ausmaß der Nichterfüllung von Anforderungen aufgrund der vorhandenen Beschränkungen
    - Entwicklungs- oder Testumgebung nicht rechtzeitig fertig
    - Ausführung, Code, Konfigurations- und Testdaten und Tests qualitativ schlecht
- Anbieterprobleme:
    - Ausfall eines Drittanbieters (z. B. Komponenten nicht rechtzeitig geliefert)
    - Vertragsprobleme

Produktrisiken (K2)

Potenzielle Fehlerbereiche (nachteilige Ereignisse oder Gefahren in der Zukunft) in der Software oder im System werden als Produktrisiken bezeichnet, weil sie ein Risiko für die Qualität des Produkts darstellen. Dazu gehören:

- Höheres Fehlerrisiko bei gelieferter Software (Software oder System kann eine erforderliche Funktion nicht innerhalb der angegebenen Grenzen ausführen)
- Softwaredokumentation qualitativ schlecht (unvollständig, inkonsistent, schwierig zu pflegen)
- Die Software/Hardware könnte potenziell einer Person oder einem Unternehmen einen Schaden zufügen
- Schlechte Softwareeigenschaften (z. B. Funktionalität, Zuverlässigkeit, Benutzerfreundlichkeit oder Leistung)
- Schlechte Datenintegrität und -qualität (z. B. Probleme bei Datenmigration, Datenkonversion, Datentransport sowie Verletzung der Datenstandards)
- Software, die die beabsichtigten Funktionen nicht ausführt und die Bedürfnisse der Stakeholder nicht erfüllt
- Geschäftsrisiko basierend auf schlechter Qualität

3.2.3 Risikomanagement (K2)

Unter Risikomanagement versteht man den Prozess der Identifizierung, Beurteilung und Priorisierung, der Planung der Reaktion auf Risiken und der Behebung und Überwachung von Risiken. Es ermöglicht die Erkennung der potenziellen Faktoren, die möglicherweise negative Auswirkungen auf die Durchführung eines Projekts und die Vorbereitung der entsprechenden Aktionen zur Behandlung von Risiken bei deren Auftreten haben.

Verschiedene Risiken können von verschiedenen Gruppen von Stakeholdern kommen. Das Entwicklungsteam sieht beispielsweise andere Risiken als die Stakeholder im Unternehmen oder die Endbenutzer.

Das Risikomanagement umfasst die folgenden Aktivitäten [ISTQB]:

- Identifizierung des Risikos
- Risikoanalyse
- Risikomigration

Behandlung potenzieller Risiken

Die Methoden zum Umgang mit Risiken können in vier Hauptkategorien unterteilt werden:

- Vermeidung
- Reduzierung
- Freigabe
- Beibehaltung

Risikomanagementplan

Ein Risikomanagementplan sollte vor und nach der Erstellung des Projektplans erstellt (und periodisch aktualisiert) werden. Der Risikomanagementplan sollte effektive Sicherheitskontrollen für den Umgang mit den Risiken angeben und einen Plan zur Kontrollumsetzung sowie die für diese Aktionen verantwortlichen Personen enthalten.

Der Risikomanagementplan umfasst:

- Liste der Risiken
- Wahrscheinlichkeit des Auftretens und/oder Priorität
- Schweregrad der Auswirkungen für jedes Risiko (einschließlich der Kosten, falls zutreffend)
- Strategien zur Beherrschung der einzelnen Risiken (einschließlich der Person/Gruppe, die für die Maßnahmen verantwortlich ist)
- Risikoeinstufungsmatrix

3.2.4 Failure Mode and Effects Analysis (Fehlermöglichkeits- und Einflussanalyse, FMEA) (K2)

Eine übliche Technik für das Risikomanagement (Identifizierung, Analyse und Planung der Reaktion) ist die Fehlermöglichkeits- und Einflussanalyse (Failure Mode and Effects Analysis) (FMEA).

Mithilfe der FMEA können für potenzielle Fehler entsprechend des Schweregrads der Konsequenzen, der Häufigkeit des Auftretens und des Grads der Erkennbarkeit Prioritäten vergeben werden. Die FMEA dokumentiert auch das aktuelle Wissen und die Maßnahmen zu den Fehlerrisiken zur Verwendung im ständigen Verbesserungsprozess. In den meisten Fällen wird die FMEA in der Entwicklungsphase des Projekts eingesetzt und ihr Hauptzweck besteht in der Vermeidung von Fehlern in der Zukunft. In späteren Phasen kann sie zur Prozesssteuerung verwendet werden. Mit der FMEA sollte schon früh in den Konzeptphasen des Projekts begonnen und sie sollte über den gesamten Lebenszyklus hinweg fortgesetzt werden. Idealerweise sollte die FMEA bereits geplant werden, sobald erste vorläufige Informationen verfügbar sind.

Die Ergebnisse einer FMEA sind Maßnahmen zur Verhinderung oder Verminderung des Schweregrads oder der Wahrscheinlichkeit von Fehlern.

Schritte zur Implementierung der FMEA

Eine FMEA wird in drei Hauptschritten entwickelt:

- Schritt 1: Bedeutung (Identifizieren der Bedeutung eines potenziellen Fehlers)
- Schritt 2: Auftretenswahrscheinlichkeit (Identifizieren, wie häufig ein potenzieller Fehler auftreten kann)
- Schritt 3: Entdeckungswahrscheinlichkeit (Identifizieren der Methoden zur Entdeckung des Fehlers)

4 Verantwortlichkeiten und Rollen (K2)	50 Minuten

Lernziele für das Foundation Level des Requirements Engineering
Die Lernziele legen fest, was Sie nach Beenden des jeweiligen Moduls gelernt haben sollten.

4.1 Basisrollen (K1)

LO-4.1.1 Die Basisrollen wiedergeben, die im Requirements Engineering maßgeblich sind (K1)

LO-4.1.2 Den Zweck und die Rolle eines Stakeholders beschreiben (K2)

4.2 Aufgaben des Requirements Engineering (K2)

LO-4.2.1 Die Aufgaben des Requirements Engineering beschreiben (K2)

LO-4.2.2 Die Eigenschaften eines „Professional for Requirements Engineering“ wiedergeben (K1)

4.1 Basisrollen (K1)	20 Minuten

Begriffe:

Kunde, Auftragnehmer, Stakeholder

4.1.1 Basisrollen (K2)

Rollen, die das Requirements Engineering betreffen oder davon betroffen sind

Kunde

Eine Person, Gruppe oder Organisation, die eine Lösung anfordert.

Auftragnehmer (Lieferant, Anbieter)

Eine Person, Gruppe oder Organisation, die die Lösung bereitstellt.

Der Kunde formuliert seine Bedürfnisse und nennt die ersten Geschäftsanforderungen und Erwartungen. Normalerweise erfolgt dies zusammen mit der Angebots-/Serviceanforderung. Es liegt in der Verantwortung des Anbieters, diese Bedürfnisse genau kennenzulernen und auf dieser Basis Anforderungen zu extrahieren.

Der Auftragnehmer liefert Lösungen auf Basis der Bedürfnisse des Kunden.

Rollen im Requirements Engineering

Requirements Manager

Ein Requirements Manager (Anforderungsmanager) ist eine Person, die für die Dokumentation, Analyse, Verfolgung, Priorisierung und Vereinbarung von Anforderungen verantwortlich ist anschließend die Änderungen steuert und die Kommunikation mit den relevanten Stakeholdern übernimmt.

Requirements Developer

Ein Requirements Developer (Anforderungsentwickler) ist eine technisch orientierte Person, die hauptsächlich mit der Erhebung, Analyse und Priorisierung von Anforderungen beschäftigt ist.

4.1.2 Stakeholder (K2)

Unter einem Stakeholder versteht man eine Gruppe oder Einzelperson, die von dem Ergebnis einer Unternehmung betroffen oder auf gewisse Weise dafür verantwortlich ist. Projekt-Stakeholder sind Einzelpersonen und Organisationen, die aktiv am Projekt beteiligt sind oder deren Interessen durch die Projektausführung oder den Projektabschluss betroffen sein können.

Stakeholder können entweder natürliche Personen, juristische Personen oder abstrakte Personen sein.

Stakeholder haben oft völlig unterschiedliche Interessen. Dies führt oft zu widersprüchlichen Anforderungen. Interessenskonflikte in den Anforderungen müssen in der Phase der Anforderungsanalyse behoben werden.

Es kann viele Kategorien von Stakeholdern geben:

- Kunden
- Endbenutzer
- Manager
- Personen, die an den organisatorischen Prozessen beteiligt sind
- Ingenieure, die für die Systementwicklung und -wartung zuständig sind
- Kunden der Organisation, die das System nutzen
- Externe Körperschaften (Behörden)
- Domänenexperten

Typische Stakeholder sind:

- Kunde
- Endbenutzer
- Projektmanager
- Produktmanager
- Systemanalyst
- Business-Analyst
- Geschäftsbeauftragte
- Marketing- und Vertriebspersonal
- Softwareentwickler
- Qualitätssicherungspersonal
- Technische Experten (Architekten, Datenbankingenieure)

- Änderungsmanager
- Kernteam des Projekts
- Management-Team

Die Identifizierung aller Stakeholder ist notwendig, um alle Sichtweisen zur geplanten Lösung angemessen berücksichtigen zu können.

Für Schulungsunternehmen: Beschreibung des typischen Stakeholder (z. B. Geschäftsführer, Projektmanager, Kunde)

4.2 Aufgaben des Requirements Engineering (K2)	30 Minuten

4.2.1 Aufgaben des Requirements Engineering (K2)

Hauptaufgaben des Requirements Engineering:

- Analyse der Geschäftsprozesse, die innerhalb einer Organisation durchgeführt werden
- Identifizierung und Analyse der Anforderungen
- Strukturierung und Modellierung der Anforderungen
- Qualitätssicherung der Anforderungen und Spezifikationen
- Erstellung der Anforderungsspezifikation
- Risikoanalyse (im Kontext der Anforderungen)
- Änderungsmanagement der Anforderungen
- Vereinbaren der Anforderungen mit den Stakeholdern

4.2.2 Kenntnisse eines „Professional for Requirements Engineering“ (K1)

Abgesehen von den technischen Kenntnissen muss der "Professional for Requirements Engineering" über die folgenden Sozialkompetenzen verfügen:

- Begabung zum Moderieren
- Selbstbewusstes Auftreten
- Überzeugungskraft
- Sprachkenntnisse
- Fähigkeit zur Kommunikation
- Genauigkeit
- Analytischer, klarer Verstand
- Fähigkeit zum strukturierten Handeln
- Methodenkompetenz
- Stressresistenz

5 Identifizierung von Anforderungen (K2)	150 Minuten

Lernziele für das Foundation Level des Requirements Engineering
Die Lernziele legen fest, was Sie nach Beenden des jeweiligen Moduls gelernt haben sollten.

5.1 Kunde (K1)

LO-5.1.1 Den Inhalt eines Vertrags wiedergeben (K1)

LO-5.1.2 Identifizieren, was bei der Bewertung von Anforderungen berücksichtigt werden sollte (K2)

5.2 Projektvisionen und -ziele (K2)

LO-5.2.1 Die Eigenschaften einer typischen Projektvision erklären (K2)

5.3 Identifizieren der Stakeholder (K2)

LO-5.3.1 Erklären, wie Stakeholder identifiziert werden können (K2)

LO-5.3.2 Stakeholder für ein bestimmtes Projekt identifizieren (K3)

5.4 Methoden zur Identifizierung von Anforderungen (K2)

LO-5.4.1 Die Ziele der Identifizierung von Anforderungen erkennen (K2)

LO-5.4.2 Verschiedene Methoden zur Identifizierung von Anforderungen anwenden (K3)

5.5 Funktionale und nicht funktionale Anforderungen (K2)

LO-5.5.1 Die Eigenschaften von funktionalen und nicht funktionalen Anforderungen beschreiben (K2)

LO-5.5.2 Die Unterschiede zwischen funktionalen und nicht funktionalen Anforderungen vergleichen (K2)

5.6 Beschreibung der Anforderungen (K2)

LO-5.6.1 Den Inhalt eines Standard-Anforderungsdokuments beschreiben (K2)

LO-5.6.2 Die Eigenschaften einer guten Anforderung beschreiben (K2)

LO-5.6.3 Eine Anforderung zusammenstellen (K3)

5.1 Kunde (K1)	20 Minuten

Begriffe:

Kunde, Vertrag

5.1.1 Kunde (K1)

Ein Kunde ist eine Organisation oder Person, die die Software kauft und einer der Schlüssel-Stakeholder des Projekts ist. Die Bedürfnisse des Kunden müssen erfüllt werden.

Der Kunde muss immer einbezogen werden. Ziel ist es, den Kunden zu verstehen und ein gegenseitiges Verständnis zu entwickeln. Der Auftragnehmer (Anbieter) sollte sich dabei immer in die Lage des Kunden versetzen.

Bei der Bewertung der Anforderungen zum Zweck der Projektplanung (was eines der Themen ist, die vom Projekt abgedeckt werden sollen) müssen verschiedene Sichtweisen berücksichtigt werden, weil eine bestimmte Anforderung für unterschiedliche Stakeholder auch unterschiedliche Prioritäten und Bedeutungen haben kann.

5.1.2 Vertrag (K2)

Die Vereinbarung (Vertrag) sollte den Wunsch des Kunden formal spezifizieren und beschreiben. Es muss sichergestellt werden, dass die Interessen des Kunden im Mittelpunkt stehen (d. h. der Anbieter erzwingt nicht eine Lösung, die er selbst bevorzugt, sondern analysiert die Bedürfnisse des Kunden und empfiehlt eine Lösung, die diese Bedürfnisse bestmöglich erfüllt).

Die Vereinbarung:

- Muss mit den verfügbaren Ressourcen zur Implementierung der Lösung konform sein
- Basiert auf: Einschätzungen, Fristen, Preisen und Projektplänen

Das Requirements Engineering liefert die Informationen für diese Einschätzungen.

Der Vertrag sollte enthalten:

- Kurze Beschreibung der geplanten Lösung
- Die Liste der priorisierten abstrakten Anforderungen
- Die Abnahmekriterien für jede Anforderung

- Die Liste der Produkte (Dokumentation, Code, funktionierende Software)
- Fristen für die Entwicklung und Lieferung des Produkts
- Andere Bedürfnisse und Erwartungen wie die bevorzugte Technik, Ressourcenanforderungen etc.

5.2 Projektvisionen und -ziele (K2)	20 Minuten

Begriffe:

Ziel, Vision

5.2.1 Vision (K2)

Die Entwicklung von Projektvisionen ist der erste Schritt im Requirements Engineering.

Die Vision sollte:

- Die Kunden, Märkte und Mitbewerber definieren
- Die zu erreichenden Ziele definieren
- Ein allgemeines Verständnis aller Stakeholder ermöglichen

Eine klare Definition der Projektvisionen ist absolut wichtig.

Für Schulungsunternehmen: Präsentation typischer Projektvisionen

Wichtige Fragen zu Projektvisionen (K2):

- Was soll das Projekt ändern?
- Wieso ist das Projekt nötig?
- Was passiert, wenn das Projekt abgeschlossen ist?
- Wer profitiert von dem Projekt?
- Welche Kosten sind wir bereit zu tragen?
- Welche Risiken wollen wir eingehen?

Für jedes Projekt muss die Vision neu festgelegt werden.

Einflüsse auf die Vision des Projekts (K1)

Die folgenden Faktoren können Einfluss auf die Vision haben:

- Kunden
  - Ziele des Kunden
  - Vorlieben des Kunden
- Strategie
  - Strategie einer Organisation
  - Marktpositionierung
  - Zu befolgende Anweisungen
- Wettbewerb
  - Wettbewerbsdaten
  - Marktentwicklung
- Produkte
  - Innovationsstufe
  - Zielgruppe
- Technologien
  - Neue Werkzeuge
  - Neue Standards
- Verfügbare Ressourcen
  - Zeit, Mitarbeiter, Kapazitäten

5.3 Identifizieren der Stakeholder (K2)	20 Minuten

Begriffe:

Stakeholder

5.3.1 Identifizieren der Stakeholder (K2)

Alle Stakeholder auf Seiten des Kunden und des Lieferanten müssen identifiziert werden.

Jeder Stakeholder oder jede Gruppe von Stakeholdern kann neue Anforderungen nennen und die Ausführung der geplanten Lösung beeinflussen. Wenn nicht alle Stakeholder identifiziert werden, besteht das Risiko, dass einige Anforderungen oder Beschränkungen unbekannt bleiben und in der Ausführung nicht berücksichtigt werden. Wenn Stakeholder fehlen, kann dies zu einem späteren Zeitpunkt im Projekt oder nach Freigabe des Systems in der Produktionsumgebung Anforderungen komplexer Änderungen an der Software nach sich ziehen

Einige der Stakeholder können Interessensgruppen bilden (zum Beispiel alle Geschäfts-Stakeholder). Interessensgruppen sollten zusammengestellt werden, weil dadurch ihre Anforderungen viel effizienter behandelt werden können.

5.3.2 Das Verfahren zur Identifizierung und Evaluierung von Stakeholdern (K2)

Das Verfahren zur Identifizierung und Evaluierung von Stakeholdern umfasst die folgenden Aktivitäten:

- Identifizierung von Stakeholdern (Analyse von Geschäftsprozessen, Bestimmen der Prozess- und Produkteigentümer, Analyse der organisatorischen Struktur und des Markts)
- Gruppieren der Stakeholder (falls möglich)
- Festlegen von Beziehungen
- Identifizierung potenzieller Konflikte
- Analyse von Konflikten und deren Ursachen und Identifizieren von Win-Win-Chancen
- Identifizierung der risikominimierenden Stakeholder, um sie mehr in die Projektaktivitäten einbeziehen zu können
- Identifizierung der Sichtweisen der Stakeholder

Für Schulungsunternehmen: Erklärung der Identifizierung und Evaluierung von Stakeholdern

5.4 Methoden zur Identifizierung von Anforderungen (K2)	40 Minuten

Begriffe:

Lernen vom Kunden, Brainstorming, Kundenbeauftragter, Feldbeobachtung, Interview, Fragebogen, Wiederverwendung, Selbstaufzeichnung, Workshop

5.4.1 Bedeutung der Identifizierung von Anforderungen (K2)

Hauptgründe für die Identifizierung von Anforderungen:

- Identifizieren aller gewünschten Funktionen, Eigenschaften, Beschränkungen und Erwartungen
- Ausrichten der Anforderungen an der Projektvision
- Detaillierte Aufgliederung der abstrakten Anforderungen und klare Beschreibung der Funktionen und Services
- Ausschließen von Funktionen und Features, die der Kunde nicht will

5.4.2 Methoden (K1)

Häufigste Methoden zur Identifizierung von Anforderungen:

- Fragebögen
- Interviews
- Selbstaufzeichnung
- Kundenbeauftragter vor Ort
- Identifizierung auf Basis vorhandener Dokumente
- Wiederverwendung (Wiederverwenden der Spezifikation eines bestimmten Projekts)
- Brainstorming
- Feldbeobachtung
- Lernen vom Kunden

- Workshops

5.4.2.1 Fragebögen

Ein Fragebogen kann aus offenen oder geschlossenen Fragen bestehen. Bei einer offenen Frage muss der Befragte seine Antwort selbst formulieren. Bei einer geschlossenen Frage wird der Befragte aufgefordert, eine Antwort aus einer Reihe möglicher Optionen auszuwählen. Diese Optionen sollten sich gegenseitig ausschließen.

Vorteile:

- Geringe Kosten
- Größere Zielgruppe

Nachteile:

- Nicht geeignet zum Sammeln von implizitem Wissen
- Niedrige Rücklaufrate ohne Motivation des Befragten
- Fragebögen können oft Anweisungscharakter haben, was die Identifizierung der echten Kundenbedürfnisse verhindert

5.4.2.2 Interview

Das Interview ist eine Gesprächsmethode, bei der der Interviewer den Befragten bittet, Informationen zu einem angegebenen Thema zu geben. Diese Methode ist in hohem Maß interaktiv und ermöglicht die Änderung der Reihenfolge vorher vorbereiteter Fragen entsprechend der Antworten und der Situation des Befragten.

Gute Interviews sind schwieriger durchzuführen als man glauben möchte, weil das normale Gesprächsverhalten eventuell im Wege steht (z. B. das Beenden der Sätze für den Gesprächspartner), was dazu führen kann, dass Interpretationen in die Daten mit einfließen. Der Interviewer sollte offene Fragen stellen, um Informationen abzufragen. Geschlossene Fragen können eingesetzt werden, um einen Status zu bestätigen (z. B. Bestätigung bereits identifizierter Anforderungen).

Vorteile:

- Der Verlauf kann an den jeweiligen Befragten angepasst werden

Nachteile:

- Zeitaufwändig
- Keine Wiederholbarkeit der Ergebnisse (Schwierigkeit, dieselben Antworten bei Wiederholen des Interviews zu erhalten)

5.4.2.3 Selbstaufzeichnung

Bei dieser Methode dokumentiert der Stakeholder (beispielsweise ein Endbenutzer) seine Aktivitäten zur Ausführung einer bestimmten Aufgabe.

Zusätzlich zur Dokumentation der Aktivitäten an sich beschreibt der Benutzer auch Änderungen, Wünsche und Bedürfnisse.

Bei diesem Ansatz ebenfalls angewendete Methoden: Demonstrationen oder Dokumenten-Reviews.

Vorteile:

- Geringer Zeitaufwand für den Requirements Engineer auf Seiten des Softwareanbieters

Nachteile:

- Vernachlässigung „automatisierter" Aktivitäten (wie Drucken und Erhalten der gedruckten Kopie)
- Hohe Abhängigkeit von der Motivation und Erfahrung des Benutzers

5.4.2.4 Kundenbeauftragter vor Ort

Dieser Ansatz ist einer der effektivsten Methoden zur Identifizierung (und Validierung) von Anforderungen, weil der Beauftragte dabei den Fortschritt systematisch überwachen und die Richtigkeit der Ausführung überprüfen und, wann immer nötig, ein Feedback und weitere Informationen geben kann.

Beauftragte des Kunden vor Ort zu haben ist eine der Hauptregeln bei agilen Methoden.

Vorteile:

- Schnelles Feedback
- Ergibt benutzerorientierte Anforderungen, die leicht angenommen werden

Nachteile:

- Hohe Kosten für den Kunden
- Anpassungskosten

5.4.2.5 Identifizierung von Anforderungen auf Basis vorhandener Dokumente

Diese Methode kann eingesetzt werden, falls bereits eine Dokumentation vorhanden ist, die bei der Identifizierung der Anforderungen in einer Organisation helfen kann. Mögliche Dokumentationen:

- Prozessmodelle und -abbildungen
- Prozessbeschreibungen
- Organigramme

- Produktspezifikation (als Ergebnis eines bestimmten Prozesses)
- Abläufe (d. h. Arbeitsabläufe)
- Normen und Anweisungen

Die identifizierten Anforderungen sind die Basis für die weitere Anforderungsanalyse und müssen mit anderen, verwandten und verknüpften Anforderungen detailliert ausgearbeitet und erweitert werden.

Vorteile:

- Keine Funktionalität wird vernachlässigt

Nachteile:

- Teuer
- Nicht geeignet, wenn es keine oder nur grundlegende Dokumente in einer Organisation gibt
- Nicht geeignet, wenn die Dokumentation nicht korrekt gepflegt ist (nicht auf dem aktuellen Stand ist)

5.4.2.6 Wiederverwendung (Wiederverwenden der Spezifikation eines bestimmten Projekts)

Die Spezifikation eines bestimmten Projekts kann wiederverwendet werden, wenn eine Organisation bereits eine oder mehrere Projekte abgeschlossen hat, die dem aktuellen Projekt ähnlich sind. Die für frühere Projekte erstellte Anforderungsspezifikation kann in einem anderen Projekt verwendet werden, um die Dauer der Anforderungsanalyse und Dokumentationserstellung zu verkürzen, wodurch früher mit der Implementierung begonnen werden kann.

In den meisten Fällen können nur Teile der vorhandenen Spezifikation in einem neuen Projekt verwendet werden. Die Dokumentation, die wiederverwendet werden soll, sollte immer dahingehend geprüft werden, ob sie mit den aktuellen Bedürfnissen und Anforderungen konform ist, und sollte sorgfältig angepasst werden.

Vorteile:

- Kosteneinsparung

Nachteile:

- Hohe Kosten bei dem ersten Projekt
- Die Wiederverwendung von Anforderungen kann ein aufwändiges und kostenintensives Änderungsmanagement erfordern, wenn diese in früheren Projekten nicht ordnungsgemäß erstellt wurden

5.4.2.7 Brainstorming

Brainstorming ist eine allgemein angewendete Methode zur Ermittlung von Anforderungen, die sich auf eher unbekannte oder neue Bereiche der Aktivität oder geplanten Systemfunktionalität einer Organisation beziehen. Anhand dieser Methode können innerhalb kürzester Zeit und mit geringem Kostenaufwand viele Ideen von verschiedenen Stakeholdern gesammelt werden. Während der Brainstorming-Sitzung übermitteln die Teilnehmer Ideen und Konzepte zu einem vorliegenden Problem.

Vorteile:

- Geringe Kosten
- Die Gelegenheit zum Sammeln vieler wertvoller Ideen in kürzester Zeit

Nachteile:

- Schwierig bei nicht motivierten Teilnehmern
- Schwierig bei weit verstreuten Teams

5.4.2.8 Feldbeobachtung

Die Feldbeobachtung ermöglicht die Beobachtung der Aktivitäten und Prozesse der Benutzer in der Ausführung und die Identifizierung der Systemanforderungen auf dieser Grundlage. Bei der Beobachtung vor Ort können die Benutzer bei der Arbeit erlebt und der Prozess, die Aufgaben und Ergebnisse dokumentiert werden. In einigen Fällen wird die Beobachtung durch Interviews der Benutzer zu ihren Jobs und die Art der Ausführung ihrer Aufgaben erweitert.

Vorteile:

- Möglichkeit zur Beobachtung der Benutzer bei der Arbeit und zur Identifizierung der tatsächlichen Anforderungen
- Nützlich, wenn die Stakeholder Probleme haben, ihre Bedürfnisse auszudrücken

Nachteile:

- Ausnahmen werden möglichweise übersehen
- Nicht geeignet in einigen Situationen (beispielsweise aus sicherheitsbezogenen oder rechtlichen Gründen)

5.4.2.9 Lernen vom Kunden

Der Zweck des Lernens vom Kunden ist es, die Anforderungen vom Kunden abzufragen, insbesondere wenn die von den Mitarbeitern des Kunden ausgeführten Prozesse und Aktivitäten mit anderen Methoden wie Interviews nicht leicht zu beschreiben sind, oder wenn der Kunde die Anforderungen zur geplanten Software nicht gut artikulieren kann.

Diese Methode stellt einen Prozess dar, bei dem der Job des Kunden direkt vor Ort erlernt wird. Der Kunde, der am besten weiß wie ein bestimmter Job ausgeführt werden muss, unterrichtet den Requirements Engineer - wie bei einem Meister und seinem Schüler.

Vorteile:

- Ist hilfreich, wenn die Mitarbeiter des Kunden Probleme haben, Dinge abstrakt zu sehen und ihre Aufgaben in Worte zu fassen.

Nachteile:

- Hohe Kosten und hoher Zeitaufwand
- Nicht geeignet in gefährlichen Umgebungen

5.4.2.10 Workshops

Workshops sind eine Art von Meetings, die sich auf spezifische (vorher definierte und den Teilnehmern angekündigte) Themen konzentrieren. Normalerweise werden dabei die Stakeholder einbezogen, die unterschiedliche Bereiche und/oder Domänen für eine kurze, intensive Zeit repräsentieren.

Workshops können verschiedene Ziele haben:

- Identifizieren von Anforderungen (zur Erstellung des Umfangs einer Lösung)
- Erkennen versteckter Anforderungen (Anforderungen, die nicht direkt benannt sind oder sogar von den Stakeholdern nicht erkannt werden, doch zur Erfüllung einiger ihrer Bedürfnisse oder abstrakter Anforderungen gebraucht werden)
- Entwickeln von Anforderungen in einem neu identifizierten Bereich (im Detail)
- Priorisieren von Anforderungen
- Schaffen eines Konsens zu den Anforderungen, wenn eine Vereinbarung zu den Anforderungen getroffen werden muss (sie abgezeichnet werden sollen)
- Überprüfen der Ergebnisse eines spezifizierten Prozesses oder einer Aktivität (Überprüfen der Spezifikation einer funktionalen Anforderung)

Vorteile:

- Bezieht Personen mit ein, die unterschiedliche Ansichten über ein vorliegendes Problem haben
- Ermöglicht die Ermittlung und Beschreibung von Anforderungen, die aus verschiedenen Sichtweisen stammen
- Ermöglicht die schnelle Entdeckung und Lösung potenzieller Konflikte zwischen den Anforderungen der Stakeholder

Nachteile:

- Schwierig im Fall von geografisch verstreuten Teams

- Verfügbarkeit aller Personen, die am Workshop teilnehmen sollen
- Bei einem Workshop ist ein Konsens nicht unbedingt leicht zu erreichen und die Diskussionen können bei (weniger wichtigen) problematischen Problemen stecken bleiben, wodurch der Prozess langwierig wird und die Teilnehmer demotiviert werden.

5.5 Funktionale und nicht funktionale Anforderungen (K2)	20 Minuten

Begriffe:

Funktionale Anforderungen, nicht funktionale Anforderungen

Um die Prüfung zu bestehen, sollten die Studierenden in der Lage sein, Beispiele funktionaler und nicht funktionaler Anforderungen zu nennen und die individuellen Qualitätskennzeichen im Detail zu beschreiben.

5.5.1 Funktionale Anforderungen (K2)

Funktionale Anforderungen geben an, was das System tut. Sie spezifizieren die Funktionen des Systems, wie es vom Endbenutzer wahrgenommen wird. Funktionale Anforderungen beschreiben auch die Auslöser des Prozesses (Benutzeraktion, Eingabe/Ausgabe von Daten, die den Start des Geschäftsprozesses verursachen).

Funktionale Anforderungen sollten durch die folgenden Qualitätskennzeichen charakterisiert sein [ISO/IEC 25000] (K1):

- Tauglichkeit
- Genauigkeit
- Interoperabilität
- Funktionalität
- Compliance
- Sicherheit

5.5.2 Nicht funktionale Anforderungen (K2)

Nicht funktionale Anforderungen geben an, wie das System arbeitet. Sie beschreiben die Qualitätsattribute des gesamten Systems oder dessen spezifische Komponente oder Funktion. Sie beschränken die Lösung dadurch, dass sie bestimmte Effizienzparameter erforderlich machen.

Nicht funktionale Anforderungen sind schwer zu beschreiben, daher werden Sie oft nur vage formuliert oder gar nicht dokumentiert. Dadurch sind Sie schwer zu testen. Aus diesem Grund sollte in allen Phasen des RE-Prozesses besondere Aufmerksamkeit auf nicht funktionale Anforderungen gelegt werden.

Aufgrund der Probleme bei der Formulierung von nicht funktionalen Anforderungen sind sie möglicherweise schwer zu testen. Nicht funktionale Anforderungen sollten daher klar formuliert werden und messbar sein.

Nicht funktionale Anforderungen [ISO/IEC 25000] (K1):

- Zuverlässigkeit
- Benutzerfreundlichkeit
- Effizienz
- Änderbarkeit
- Mobilität

Nicht funktionale Anforderungen geben Kriterien an, die zur Beurteilung des Betriebs eines Systems verwendet werden können. Daher haben Sie einen großen Einfluss auf die Zufriedenheit des Kunden bei Verwendung der Software. Funktionale Anforderungen müssen Funktionen angeben; nicht funktionale Anforderungen legen fest, wie leicht und effizient die Funktionen verwendet werden können.

5.6 Beschreibung der Anforderungen (K2)	30 Minuten

5.6.1 Beschreibung der Anforderungen (K2)

Anforderungen müssen klar und genau spezifiziert werden. Sie sollten messbar sein, um sicher zu stellen, dass sie getestet werden können und dass ihre Implementierung ordnungsgemäß überprüft werden kann. Man sollte immer vor Augen haben, dass die Gemeinsprache Grenzen und Nachteile hat. Daher können Beschreibungen von Anforderungen unklar und zweideutig sein. Aus diesem Grund sollten, wo immer möglich, Normen und Vorlagen verwendet werden. Normen liefern ein allgemeines Verständnis und die bewährten Vorgehensweisen der Spezifikation, während Vorlagen die zu verwendende Sprache einschränken.

Zusätzlich zu Normen und Vorlagen sind Wortsammlungen ein wichtiges Werkzeug zur Vereinfachung der Kommunikation zwischen unterschiedlichen Stakeholdern und zur Einführung einer gewissen Kontrolle über die Zweideutigkeit der natürlichen Sprache.

Die Beschreibung einer Anforderung muss verschiedene Kriterien erfüllen (z. B. klar, genau, unzweideutig, messbar, ohne überflüssige Wörter etc.).

5.6.2 Verfahren zur Erstellung von Anforderungen (K3)

Anforderungen werden in den folgenden Schritten erstellt:

1. Festlegen des betroffenen Prozesses
   - Fokus auf Funktionalität
   - Festlegen der Eingaben und Ausgaben
2. Klassifizierung der Systemaktivität
   - Identifizieren der unabhängigen Systemaktivität
   - Identifizieren der Interaktionen mit dem Benutzer
   - Identifizieren der Anforderungen an die Oberfläche
3. Ermittlung der rechtlichen Verpflichtung
   - Klare Darstellung der rechtlichen Verpflichtung durch Schlüsselwörter (sollte, muss etc.)
4. Verfeinern der Beschreibung des Prozesses
   - Detaillierte Beschreibung von Objekten und Integrationspunkten

5. Logische und zeitliche Beschränkungen
   - Erstellung der Rahmenbedingungen

Beispiel: Eine Anforderung für die Generierung einer Rechnung anhand der Daten aus einem externen System.

Verfahrensschritt	Ausgabe – Beschreibung einer Anforderung
1. Festlegung des Prozesses: • Der Systemprozess der Generierung einer Rechnung	Das System generiert eine Rechnung
2. Klassifizierung der Systemaktivität • Generieren einer Rechnung nach dem Auftrag des Kunden • Schnittstelle zu einem externen System wird eingerichtet	Nachdem der Benutzer eine Rechnungsanforderung gesendet hat, generiert das System eine Rechnung anhand der Daten aus dem externen System.
3. Ermittlung der rechtlichen Verpflichtung • Das System muss eine Rechnung anhand der korrekten Daten aus dem externen System generieren	Nachdem der Benutzer eine Rechnungsanforderung gesendet hat, muss das System eine Rechnung anhand der Daten aus dem externen System generieren.
4. Verfeinern der Beschreibung des Prozesses • Benennen des externen Finanzsystems	Nachdem der Benutzer eine Rechnungsanforderung gesendet hat, muss das System eine Rechnung anhand der Daten aus dem SAP-Finanzsystem generieren.
5. Logische und zeitliche Beschränkungen • Festlegen einer zeitlichen Beschränkung – maximale Dauer des Vorgangs beträgt 30 Sekunden	Nachdem der Benutzer eine Rechnungsanforderung gesendet hat, muss das System innerhalb von 30 Sekunden eine Rechnung anhand der Daten aus dem SAP-Finanzsystem generieren.

Für Schulungsunternehmen: Beschreibung der einzelnen Schritte zur Erstellung einer Anforderung

5.6.3 Anforderungsdokument (K2)

Der Standardinhalt des Anforderungsdokuments ist [IEEE 830]:

Inhalt

1. Einführung
    - 1.1 Zweck
    - 1.2 Umfang
    - 1.3 Definitionen, Akronyme und Abkürzungen
    - 1.4 Referenzen
    - 1.5 Übersicht
2. Gesamtbeschreibung
    - 2.1 Produktperspektive
    - 2.2 Produktfunktionen
    - 2.3 Benutzereigenschaften
    - 2.4 Beschränkungen
    - 2.5 Annahmen und Abhängigkeiten
3. Spezifische Anforderungen
    - 3.1 Externe Schnittstellen
    - 3.2 Funktionen
    - 3.3 Leistungsanforderungen
    - 3.4 Logische Datenbankanforderungen
    - 3.5 Ausführungsbeschränkungen
        - 3.5.1 Normenkonformität
    - 3.6 Softwaresystemattribute
        - 3.6.1 Zuverlässigkeit
        - 3.6.2 Verfügbarkeit
        - 3.6.3 Sicherheit
        - 3.6.4 Wartbarkeit
        - 3.6.5 Mobilität

Anhang

Index

6 Spezifikation von Anforderungen (K2)	100 Minuten

Lernziele für das Foundation Level des Requirements Engineering
Die Lernziele legen fest, was Sie nach Beenden des jeweiligen Moduls gelernt haben sollten.

6.1 Spezifikation (K2)

LO-6.1.1 Den Zweck und Inhalt einer Anforderungsspezifikation beschreiben (K2)

LO-6.1.2 Die Eigenschaften einer Anforderungsspezifikation beschreiben (K2)

LO-6.1.3 Den Zweck und Inhalt einer Anforderungsspezifikation beschreiben (K2)

LO-6.1.4 Die Eigenschaften einer Lösungsspezifikation beschreiben (K2)

LO-6.1.5 Die Normen wiedergeben, die wichtig für Anforderungs- und Lösungsspezifikationen sind (K1)

LO-6.1.6 Die Bedeutung einer „User Story“ erklären (K2)

6.2 Verfahren (K3)

LO-6.2.1 Ein typisches Verfahren zur Spezifikation von Anforderungen anwenden (K3)

6.3 Formalisierung (K2)

LO-6.3.1 Die verschiedenen Grade der Formalisierung erklären, die für die Spezifikation von Anforderungen existieren (K2)

LO-6.3.2 Den spezifischen Grad an Formalisierung in einem vorliegenden Szenario anwenden (K3)

6.4 Qualität der Anforderungen (K2)

LO-6.4.1 Die Konsequenzen von Fehlern in Anforderungen wiedergeben (K1)

LO-6.4.2 Die Möglichkeiten zur Vermeidung von Anforderungsfehlern beschreiben (K2)

6.1 Spezifikation (K2)	30 Minuten

Begriffe:

IEEE 830, Anforderungsspezifikation, Lösungsspezifikation

6.1.1 Spezifikation (K1)

Eine Spezifikation ist ein eindeutiger Satz von Anforderungen, den ein Material, Produkt oder Service erfüllen muss.

Die Spezifikation dient dazu, die Anforderungen zurückzuverfolgen und zu steuern. In der Spezifikation werden die Anforderungen strukturiert angegeben und separat modelliert (Anforderungen werden “unabhängig” modelliert, weil abstrakte Anforderungen auf eine Ebene heruntergebrochen werden, auf der jede einzelne Anforderung eine "unabhängige" Entität ausmacht, die weiter entwickelt und verfolgt werden kann). Eine Spezifikation ist eine formale Vereinbarung über Anforderungen, die im geplanten Softwaresystem (oder einer anderen Form von Lösung) implementiert werden sollen.

Der Begriff „Spezifikation“ kann im Kontext von Anforderungen und Lösungen verwendet werden.

6.1.2 Anforderungsspezifikation (K2)

Die Anforderungsspezifikation beschreibt den Problembereich (einen Interessensbereich wie beispielsweise eine Lösung zu einem Geschäftsproblem, eine neue Funktion etc.) und enthält mindestens die folgenden Informationen:

- Kundenanforderungen
- Beschränkungen
- Abnahmekriterien

Die Erstellung der Spezifikation zu den Kundenanforderungen sollte Aufgabe des Kunden sein. In einigen Fällen kann der Anbieter jedoch die Anforderungsspezifikation vorbereiten.

Erstellung von anderen Arten von Spezifikationen von: Systemanforderungen, Softwareanforderungen, Sicherheitsanforderungen, Umweltanforderungen, rechtlichen Anforderungen etc. sind Aufgabe anderer Rollen.

6.1.3 User Storys (K2)

User Storys werden in agilen Softwareentwicklungsmethoden eingesetzt. User Storys sind eine schnelle Methode zur Verarbeitung von Kunden- und Benutzeranforderungen. Die User Story beabsichtigt eine schnellere Reaktion auf sich schnell verändernde Anforderungen in der Realität bei niedrigeren Gemeinkosten.

Die User Story beschreibt die Funktionalität, die den Wert ausmacht. User Storys setzen sich aus drei Aspekten zusammen:

- Eine schriftliche Beschreibung der Story, die zur Planung und Erinnerung verwendet wird
- Gespräche über die Story, die dazu dienen, die Details der Story herauszukristallisieren
- Tests, die Details vermitteln und dokumentieren und verwendet werden können, um zu ermitteln, wann eine Story abgeschlossen ist [Mike Cohn: User Stories applied. 2009]

6.1.4 Lösungsspezifikationen (K2)

Lösungsspezifikationen werden auch als funktionale Spezifikationen, Systemanforderungsspezifikationen oder Softwareanforderungsspezifikationen bezeichnet und beschreiben den Lösungsbereich.

Eine funktionale Spezifikation ist ein Dokument, das klar die technischen Anforderungen für die Lösung beschreiben. Die funktionale Spezifikation ist die Basis für die Weiterentwicklung des Systems. Daher muss sie präzise Informationen über alle funktionalen Aspekte der zu implementierenden Software angeben. Auf dieser Basis können Architekten und Entwickler die technischen Aspekte des Systems effizient entwickeln. Die funktionale Spezifikation ist ein Leitfaden für Tester zur Verifizierung der einzelnen funktionalen Anforderungen (z. B. ist eine Spezifikation eine Testbasis).

Die funktionale Spezifikation beschreibt nicht, wie die Systemfunktionen implementiert werden und welche Technologie zu verwenden ist. Sie konzentriert sich vielmehr auf die Funktionalität, die Interaktionen zwischen dem Benutzer und der Software.

Möglicher Zweck der funktionalen Spezifikation:

- Grundlage für das allgemeine Verständnis des Umfangs und der Funktionalität der zu implementierenden Lösung
- Sicherstellen der Übereinstimmung im Team über das, was mit dem System erreicht werden soll, bevor mit dem Schreiben des Quellcodes und der Handbücher und der Vorbereitung der Daten und Tests begonnen wird.
- Bereitstellen einer detaillierten Beschreibung der notwendigen Funktionalität in Bezug auf die Interaktionen zwischen Benutzern und Software für das Entwicklungsteam
- Bereitstellen der Grundlage für das Testverfahren des Testteams

6.1.5 Wichtige Normen (K1)

Die folgenden Normen können zur Erstellung von Spezifikationen herangezogen werden:

- IEEE 1362 (Systemleistungsspezifikationen)
- IEEE 830 (Softwareanforderungsspezifikation)
- IEEE 1233 (Funktionale Systemspezifikationen)

6.2 Verfahren (K3)	30 Minuten

6.2.1 Verfahren der Lösungsspezifikation (K3)

Die Spezifikation dient als Aktivität zur Formalisierung der Ergebnisse aus der Anforderungsanalyse (K2).

Die Identifizierung, Analyse und Anforderungsspezifikationen führen zur Vereinbarung von Anforderungen (weitere Informationen hierzu finden Sie im Abschnitt).

Identifizierte, analysierte und modellierte Anforderungen sollten klar und unmissverständlich dokumentiert werden.

Ein Verfahren der Lösungsspezifikation umfasst die folgenden Aktivitäten:

1. Identifizierung der Stakeholder
2. Definition der Vision und Ziele
3. Anforderungsfestlegung
4. Strukturierte Anforderungsspezifikation
5. Beschreibung der Systemumgebung
6. Festlegen der Lösung (Definition des Systems und Anwendungsbereichs mit den relevanten Aspekten außerhalb des Anwendungsbereichs (doch mit Einfluss darauf), wie Schnittstellen zu externen Systemen)
7. Anforderungsanalyse
8. Modellierung des Problems
9. Modellierung der Lösung

Das Verfahren formalisiert die Ergebnisse des Vorgangs der Anforderungsanalyse. Das Lösungsmodell ist eine Grundlage für die Ausführung und Implementierung.

Das Verfahren bezieht eine Reihe von Stakeholdern mit ein, die die Spezifikationsarbeit in verschiedenen Bereichen unterstützen. Die Ausgabe des Verfahrens, die Lösungsspezifikation, dient als Startpunkt für die Entwicklung von Software, Hardware und Datenbank. Sie beschreibt die Funktion (funktionale und nicht funktionale Spezifikationen) des Systems, der Systemleistung und der Beschränkungen des Betriebs und der Benutzeroberfläche.

6.3 Formalisierung (K2)	20 Minuten

Begriffe:

Formale Ebene, nicht formale Ebene, halb formale Ebene

6.3.1 Grade der Formalisierung (K2)

Eine Anforderungsspezifikation kann in unterschiedlichen Formalisierungsgraden erstellt werden:

- Nicht formal
- Halb formal
- Formal

6.3.1.1 Nicht formale Ebene

Der nicht formale Ansatz zum Schreiben einer Spezifikation bedeutet, dass das Dokument in Gemeinsprache geschrieben wird, ohne formale Notation. Dieser Ansatz kann verfolgt werden, wenn die Leser keine Erfahrung mit eher formalem und technischem Spezifikationswortschatz haben und dann Schwierigkeiten hätten, den Inhalt des Dokuments zu verstehen. Die Hauptschwäche bei diesem Ansatz liegt in seiner Vieldeutigkeit und kann zu Missverständnissen und Überinterpretation führen. Darüber hinaus ist eine nicht formale Spezifikation keine gute Grundlage für die Implementierung und die Tests, weil sie nicht klar und präzise genug ist.

6.3.1.2 Halb formale Ebene

Eine halb formale Spezifikation enthält Teile einer formalen Notation und ist gut strukturiert. Normalerweise basieren diese Spezifikationen auf einer spezifischen Vorlage (oft von relevanten Normen abgeleitet). Halb formale Spezifikationen können Anforderungen in einer Form von Modellen ausdrücken und verwenden eine formalisierte Gemeinsprache.
Die am häufigsten verwendeten Notationen für die halb formale Dokumentation sind UML und SysML.

6.3.1.3 Formale Ebene

Die formale Spezifikation ist eine mathematische Beschreibung von Software, die zur Entwicklung und Implementierung verwendet werden kann. Die formale Spezifikation beschreibt, was das System tun sollte. Normalerweise beschreibt sie nicht, wie das System dies tun soll. Da sie auf mathematischen Formeln basiert und schwieriger zu erlernen ist, wird die formale Spezifikation eher selten verwendet und erfordert mathematische Kenntnisse.

6.4 Qualität der Anforderungen (K2)	20 Minuten

Begriffe:

Checkliste, Qualitätseigenschaften, Review, Validierung, Verifizierung, Rückverfolgbarkeit

6.4.1 Hintergrund

Anforderungsfehler als Ursache für hohe Kosten (K2)

Da die Anforderungen die Grundlage für die Systementwicklung bilden, zieht sich jeder Fehler und jede fehlende Anforderung durch alle anderen Entwicklungsprozesse im Projekt. Es ist wichtig zu wissen, dass Fehler aus qualitativ schlechten Anforderungen in späteren Phasen des Projekts nur mit einem erheblich höheren Kostenaufwand behoben werden können als andere Arten von Fehlern. Außerdem werden die Kosten zur Behebung der Fehler immer höher, je später sie entdeckt werden.

Daher sind die Verifizierung (produzieren wir das Produkt auf korrekte Weise) und die Validierung (produzieren wir das richtige Produkt) der Anforderungen auf jeden Fall erforderlich.

Anforderungen sollten dokumentiert und anschließend anhand der Qualitätskriterien getestet werden (weitere Informationen hierzu finden Sie in Kapitel 1.1 Anforderung.

6.4.2 Maßnahmen zur Qualitätsverbesserung und Qualitätssicherung der Anforderungen (K2)

Die folgenden Werkzeuge und Methoden können zur Qualitätsverbesserung und Qualitätssicherung der Anforderungen verwendet werden:

- Normen und Vorlagen
- Reviews und Untersuchungen
- Rückverfolgbarkeit
- Prototyping
- Beobachtung der Qualitätskriterien (Qualitätskriterien können sein: Vollständigkeit, Richtigkeit, Konformität der Anforderungsspezifikationen mit den entsprechenden Normen)

Die Qualität einer Anforderungsspezifikation kann durch Integration der folgenden Elemente verbessert werden:

- Überblick über den Zweck der Dokumente, den Anwendungsbereich, die Definitionen und ein Glossar
- Überblick über die Ziele auf verschiedenen Ebenen (d. h. eine abstrakte Anforderungsspezifikation hat andere Ziele als eine detaillierte Spezifikation einer funktionalen Anforderung)
- Definieren der Ausführungs- und Implementierungsbeschränkungen
- Abstufung/Priorisierung der Anforderungen
- Klare Aussagen darüber was ein System tun sollte und nicht wie es dies tun sollte
- Dokumentieren der Gesetzgebung, Annahmen, Geschäftsregeln und Abhängigkeiten
- Vermeiden ergänzender Beschreibungen von Diagrammen, die klar sind und für sich selbst sprechen (schwieriger, abstrakter Text sollte nach Möglichkeit durch Diagramme ersetzt werden)
- Klar angegebener Benutzerkatalog und Berechtigungsschema (Benutzerrechte und -berechtigungen)
- Strukturierte Präsentation
- Einfache, klare, präzise und unmissverständliche Sprache

7 Anforderungsanalyse (K2)	140 Minuten

Lernziele für das Foundation Level des Requirements Engineering
Die Lernziele legen fest, was Sie nach Beenden des jeweiligen Moduls gelernt haben sollten.

7.1 Anforderungen und Lösungen (K1)

LO-7.1.1 Das Ziel der Anforderungsanalyse wiedergeben (K1)

LO-7.1.2 Das Verfahren bei der Anforderungsanalyse beschreiben (K2)

LO-7.1.3 Das Konzept des strukturellen Bruchs zwischen Anforderungen und Lösungen erklären (K2)

7.2 Methoden und Techniken (K2)

LO-7.2.1 Die unterschiedlichen Modelle der Anforderungsanalyse wiedergeben (K1)

LO-7.2.2 Verschiedene Analysemethoden für spezifische Modelltypen anwenden (K3)

7.3 Objektorientierte Analyse (K2)

LO-7.3.1 Die Eigenschaften von UML beschreiben (K2)

LO-7.3.2 Die Eigenschaften von SysML beschreiben (K2)

7.4 Kostenschätzung (K2)

LO-7.4.1 Den Grund für eine Kostenschätzung wiedergeben (K1)

LO-7.4.2 Die wichtigen Faktoren für eine Kostenschätzung erkennen (K1)

7.5 Priorisierung (K2)

LO-7.5.1 Das Verfahren zur Priorisierung in einem vorgegebenen Szenario anwenden (K3)

7.6 Vereinbaren von Anforderungen (K2)

LO-7.6.1 Identifizieren, was bei der Vereinbarung von Anforderungen berücksichtigt werden sollte (K2)

7.1 Anforderungen und Lösungen (K1)	20 Minuten

7.1.1 Ziel der Anforderungsanalyse (K2)

Das Ziel der Anforderungsanalyse besteht darin, eine Lösung für die Implementierung der Anforderungen zu erstellen. Die Anforderungsanalyse berücksichtigt die verschiedenen Stakeholder der Lösung, mögliche Konflikte der Anforderungen, Analysebeziehungen und Abhängigkeiten zwischen den Anforderungen etc.

7.1.2 Verfahren der Anforderungsanalyse (K2)

Das Verfahren der Anforderungsanalyse umfasst die folgenden Schritte:

1. Analyse der Bedürfnisse
2. Beschreibung der Lösung
3. Kostenschätzung und Priorisierung

7.1.3 Struktureller Bruch zwischen Anforderungen und Lösungen (K2)

Struktureller Bruch zwischen Anforderungen und Lösungen bedeutet, dass es einen Unterschied zwischen den Anforderungen und Lösungen gibt. Eine Lösung ist eine Implementierung einer Anforderung. Das Anforderungsmodell kann als Geschäftsmodell gesehen werden, das das Geschäftsproblem darstellt, das vom Lösungsmodell behoben werden soll. Ein Lösungsmodell ist detaillierter und enthält die technische Spezifikation der Anforderungen; es ist eine Grundlage für die Entwicklung und den Testvorgang.

Es gibt drei grundlegende Modellebenen:

- Anforderungsmodell
  - Oft eine nicht formale Spezifikation
  - Notationen für die Geschäftsmodellierung (BPMN – Business Process Modeling Notation)
- Lösungsmodell
  - Funktionale Strukturen (Algorithmen, Verfahren, Workflows)

- Konzeptmodell
    - Technologische Software/Hardware-Spezifikationen

Die Modelle sollten rückverfolgbar sein.

7.2 Methoden und Techniken (K2)	20 Minuten

Begriffe:

Kontextmodell, Datenflussmodell, Entity-Relationship-Modell (Gegenstand-Beziehung-Modell, ER-Modell), funktionale Zerlegung, Konzeptmodelle, Anforderungsmodelle, Lösungsmodelle, Zustandsübergangsmodell

7.2.1 Analysemethoden und -modelle

Verschiedene Aspekte eines Systems werden durch verschiedene Ansichten dargestellt.

Modelle werden durch geeignete Analysemethoden entwickelt.

Analysemethode	Analysemodell
Kontextanalyse	Kontextmodell
Architekturanalyse	Funktionale Zerlegung
Datenstromanalyse	Datenstrommodell
Bedingungsanalyse	Bedingungsmodell
Entscheidungsanalyse	Entscheidungstabelle
Datenanalyse	Semantisches Datenmodell ER-Modell (Entity-Relationship-Modell, Gegenstand-Beziehung-Modell) Datenkatalog

7.2.2 Modelltypen (K2)

Es gibt drei grundlegende Modelltypen:

- Anforderungsmodell
  - Beschreibt den Problembereich
  - Entwickelt in frühen Projektphasen
  - Dient der Anforderungsanalyse und Kostenschätzung
  - Stellt eine Grundlage für das Lösungsmodell dar
  - Beispiel: Geschäftsanwendungsfall, Geschäftsklassemodell, Geschäftsprozessmodell
- Lösungsmodell
  - Beschreibt den Lösungsbereich aus verschiedenen Sichtweisen des Systems
  - Gleichzeitig mit dem Anforderungsmodell entwickelt
  - Legt die Form der Implementierung der funktionalen und nicht funktionalen Anforderungen fest
  - Stellt eine Grundlage für die Systemausführung dar
  - Beispiel: Anwendungsfallmodell, Sequenzdiagramm, Aktivitätsdiagramm, Zustandsübergangsdiagramm
- Konzeptmodell
  - Technologische Software/Hardware-Spezifikationen (Module, Hardwarekomponenten, PC-Eigenschaften)

7.2.3 Verschiedene Perspektiven des Systems (K2)

Einige auf das System bezogene Sichtweisen:

- Logische Sicht
  - Funktionale Anforderungen
- Prozesssicht
  - Kommunikation
  - Interaktion
  - Nicht funktionale Anforderungen
- Implementationssicht
  - Komponenten (Module)

- Installationssicht
  - Integration
  - Systemarchitektur

7.2.4 Verschiedene Modelle (K1)

Die folgenden Modelle können verwendet werden:

- Kontextmodell
  - Statische Beschreibung des Systems
  - Drückt die Basisarchitektur aus
- Funktionale Zerlegung
  - Statische Beschreibung des Systems
  - Drückt die schrittweise Zerlegung des Systems aus
- Datenflussmodell
  - Dynamische Beschreibung des Systems
  - Grafische Darstellung des Datenflusses durch ein System
  - Liefert keine Informationen über das Timing des Prozesses und darüber, ob Prozesse nacheinander oder parallel ablaufen
- Zustandsübergangsmodell
  - Dynamische Beschreibung des Systems
  - Stellt das Verhalten des Systems bei einer Reihe von Ereignissen dar, die in einem Zustand oder in mehreren Zuständen auftreten könnten
- ER-Modell (Entity-Relationship-Modell, Gegenstand-Beziehung-Modell)
  - Abstrakte und konzeptionelle Darstellung von Daten
  - Enthält Bausteine: Gegenstände, Beziehungen und Attribute

Wann sollte ein spezifisches Modell angewendet werden? Verschiedene Modelle beantworten jeweils andere Fragen zur Lösung.

- Kontextmodell ⇨ WAS (Darstellung der Hauptflüsse zwischen dem System und der Außenwelt)
- Funktionale Zerlegung ⇨ WAS (welche Funktionen und Features gehören zum Anwendungsbereich des Systems)
- Datenflussmodell ⇨ WAS (Flüsse zwischen Geschäftsprozess/ Funktionen /Aktivitäten)

- ER-Modell ⇨ WAS (welche Beziehungen bestehen zwischen spezifischen Gegenständen (Objekten) des Systems)
- Zustandsübergangsmodell ⇨ WARUM (Ursache und Wirkung)

7.3 Objektorientierte Analyse (K2)	30 Minuten

Begriffe:

Aktivitätsdiagramm, Verhaltensdiagramm, Klassendiagramm, Kommunikationsdiagramm, Komponentendiagramm, Kompositionsstrukturdiagramm, Verteilungsdiagramm, Interaktionsübersichtsdiagramm, Objektdiagramm, Objektorientierte Analyse, Paketdiagramm, Anforderungsdiagramm, Sequenzdiagramm, Zustandsautomatdiagramm, Strukturdiagramm, SysML, Zeitverlaufsdiagramme, UML, Anwendungsfalldiagramm

7.3.1 UML (K1)

Die Vereinheitlichte Modellierungssprache (Unified Modeling Language, UML) ist eine vereinheitlichte Notation für die Analyse und Ausführung von Systemen. Sie enthält verschiedene Diagrammtypen für verschiedene Sichtweisen des Systems (Struktur- oder Verhaltensdiagramme).

7.3.1.1 Verhaltensdiagramme (K2)

Verhaltensdiagramme bilden Verhaltensfunktionen eines Systems oder Geschäftsprozesses ab.

Zu diesen Diagrammen gehören die folgenden Diagrammtypen:

- Aktivitätsdiagramme
  - Modellieren das Verhalten eines Systems und die Art und Weise, in der diese Verhaltensweisen in einem Gesamtfluss des Systems in Beziehung stehen.
- Anwendungsfalldiagramme
  - Erfassen Anwendungsfälle und Beziehungen zwischen den Akteuren und dem System.
  - Beschreiben die funktionalen Anforderungen des Systems, die Art und Weise, in der externe Bediener an den Systemgrenzen interagieren, sowie das Verhalten des Systems.
- Zustandsautomatdiagramme
  - Zeigen, wie ein Element die Zustände wechselt, wobei dessen Verhalten gemäß der Übertragungsauslöser und begrenzender Schutzeinrichtungen klassifiziert wird.
- Zeitverlaufsdiagramme
  - Definieren das Verhalten verschiedener Objekte auf einer Zeitskala.

- Bieten eine visuelle Darstellung von Objekten, die den Zustand wechseln und im Lauf der Zeit interagieren.

- Sequenzdiagramme
  - Strukturierte Darstellung des Verhaltens als eine Reihe von sequenziellen Schritten im Lauf der Zeit.
  - Werden verwendet zur Abbildung des Workflows, der Nachrichtenübertragung und der Art und Weise, wie Elemente im Allgemeinen im Lauf der Zeit kooperieren, um ein Ergebnis zu erzielen.
- Kommunikationsdiagramme
  - Zeigen die Interaktionen zwischen Elementen bei der Laufzeit, wobei die Beziehungen zwischen den Objekten visualiert werden.
- Interaktionsübersichtsdiagramme
  - Visualisieren die Kooperation zwischen anderen Interaktionsdiagrammen, um einen Kontrollfluss abzubilden, der einem umfassenden Zweck dient.

7.3.1.2 Strukturdiagramme (K2)

Strukturdiagramme bilden die Strukturelemente ab, die ein System oder eine Funktion ausmachen. Diese Diagramme spiegeln die statischen Beziehungen einer Struktur wieder, wie Klassen- oder Paketdiagramme, oder Laufzeitarchitekturen wie Objekt- oder Kompositionsstrukturdiagramme.

Zu den Strukturdiagrammen gehören die folgenden Diagrammtypen:

- Klassendiagramme
  - Erfassen die logische Struktur des Systems, die Klassen und Objekte und erstellen das Modell, beschreiben was existiert und welche Attribute und Verhaltensweisen vorhanden sind.
- Kompositionsstrukturdiagramme
  - Spiegeln die interne Zusammenarbeit der Klassen, Schnittstellen und Komponenten (und deren Eigenschaften) wieder, um eine Funktionalität zu beschreiben.
- Komponentendiagramme
  - Präsentieren die Teile der Software, die ein System erstellen sowie deren Struktur und Abhängigkeiten.
- Verteilungsdiagramme
  - Beschreiben die Ausführungsarchitektur des Systems.

- Objektdiagramme
  - Präsentieren die Objektinstanzen der Klassen und deren Beziehungen zu einem bestimmten Zeitpunkt.
- Paketdiagramme
  - Bilden die Strukturierung der Modellelemente in den Paketen und die Abhängigkeiten untereinander ab.

7.3.2 SysML (K2)

Die Systems Modeling Language (SysML) ist eine spezielle Modellierungssprache für die Systemtechnik. Sie stellt eine Erweiterung von UML 2.1 dar.

SysML bietet einige Verbesserungen im Vergleich zu UML. Diese Verbesserungen sind wie folgt:

- Flexiblere und ausdrucksstärkere Semantik
  - Reduziert die softwarebezogenen Einschränkungen und fügt zwei neue Diagrammtypen hinzu: Anforderungsdiagramm und Parameterdiagramm.
  - Ermöglicht die Modellierung eines breiten Spektrums von Systemen, die Hardware, Software, Informationen, Prozesse , Personal und Einrichtungen umfassen.
- Leicht zu erlernen und anzuwenden
  - Geringerer Umfang als UML (lässt viel der softwarebezogenen Konstrukte der UML außer Acht)
- Unterstützende Modelle und Ansichten
  - Modelle und Ansichten, deren Architektur mit IEEE-Std-1471-2000 abgestimmt ist.
- Wiederverwendung von sieben der UML-Diagramme und Bereitstellung von zwei neuen Diagrammen (Anforderungsdiagramm und Parameterdiagramm), also insgesamt neun Diagrammtypen
  - Anforderungsdiagramme zur Erfassung von funktionalen Anforderungen sowie Leistungs- und Schnittstellenanforderungen
  - Parametrische Diagramme zur Definition der Leistungsbeschränkungen und quantitativen Beschränkungen

7.4 Kostenschätzung (K2)	20 Minuten

Hintergrund

Kostenschätzungen verbinden das Requirements Engineering mit dem Projektmanagement.

7.4.1 Typen von Kostenschätzungen (K2)

Die häufigsten Aspekte für die Kostenschätzung sind:

- Kosten
- Zeit
- Anforderungen

Kostenschätzungen helfen dabei, die Kosten für die Entwicklung, Änderungen etc. zu erkennen.

7.4.2 Einflüsse auf die Entwicklungskosten (K2)

Die Kosten des Projekts hängen von verschiedenen Faktoren ab:

- Projekttyp
- Prozessreife
- Design- und Testmethoden und -werkzeuge
- Technologie
- Komplexität der geplanten Lösung
- Qualitätsziele (beispielsweise die gewünschte Stufe der Softwarequalität)
- Teamqualifikationen
- Teamverteilung
- Erfahrungen

Die Exaktheit der Kostenschätzung hängt vom Fortschritt des Projekts und dessen Reifegrad ab.

7.4.3 Ansätze für die Kostenschätzung

Die Kostenschätzung kann anhand der folgenden Hilfsmittel durchgeführt werden:

- Rückschlüsse aus Analogien
- Algorithmisches Verfahren

7.4.3.1 Rückschlüsse aus Analogien (K2)

Die Kostenschätzung basiert auf dem Vergleich mit ähnlichen Kosten des Projekts. Sie basiert auf Erfahrung, nicht auf mathematischen Formeln. Diese Methode vergleicht das aktuelle Projekt mit früheren Projekten. Verglichen werden kann:

- Die Anzahl der Anforderungen
- Der Anwendungsbereich der Lösung
- Die eingesetzte Technologie
- Die Eigenschaften der Mitarbeiter (Fertigkeiten, Erfahrung)

Basierend auf den Ergebnissen des Vergleichs kann eine Kostenschätzung erstellt werden.

Kostenschätzungsverfahren basieren immer auf historischen Daten und Rahmenbedingungen.

Delphi-Methode

Die Delphi-Methode ist eine strukturierte Kommunikationstechnik zur Durchführung einer interaktiven Prognose. Dazu wird eine Gruppe von Experten befragt [Linstone75].

Die Experten werden aufgefordert, in einigen Runden Fragebögen auszufüllen. Nach jeder Runde werden eine anonyme Zusammenfassung der Prognosen dieser Experten und die Gründe für die abgegebenen Beurteilungen präsentiert. Danach überarbeiten die Experten ihre früheren Antworten und berücksichtigen dabei die Antworten der anderen Mitglieder in ihrer Gruppe.

Man glaubt, dass bei diesem Prozess die Anzahl der Antworten geringer wird und die Gruppe sich schließlich auf die "richtige" Antwort einigt.

Der Prozess wird an einem vordefinierten Stoppkriterium angehalten. Die mittleren Noten der Endrunden bestimmen das Ergebnis.

Agile Schätzungen

In agilen Projekten, die oft von Scrum gesteuert werden, gibt es eine Schätzungsmethode namens Planspiel (Planning-Game) oder Pokern (Playing Poker). Diese Methode wird eingesetzt, um im Team einen Konsens zu schaffen. Die Teamfähigkeit wird dann durch etwas gemessen, das „Burn

Down Rate“ genannt wird, und verbessert durch Rückschausitzungen, die nach jedem Sprint durchgeführt werden und in denen die Sollzahlen mit den Istzahlen verglichen werden. Dadurch kann das Team seine Fähigkeit einer Kostenschätzung verbessern.

7.4.3.2 Algorithmisches Verfahren (K2)

In diesem Ansatz werden die Kosten auf Basis von Parametern berechnet. Parameter können das Produkt (Volumen, Dauer), die Rahmenbedingungen (Effizienz) etc. beschreiben.

Die folgenden Methoden können angewendet werden:

- Putnam-Formel (Gleichung)
- Function-Points
- Constructive Cost Model (CoCoMo)

Putnam-Gleichung [Putnam91]

$$\text{Aufwand} = [\text{Größe}/\text{Produktivität} * \text{Zeit}^{4/3}]^3 * B$$

Erklärung:

- Größe – die Produktgröße (geschätzt anhand einer beliebigen Größenschätzung durch eine Organisation). Putnam verwendet ESLOC (Effective Source Lines of Code)
- B – ein Skalierungsfaktor als Funktion der Projektgröße
- Produktivität – die Prozessproduktivität, die Fähigkeit einer bestimmten Softwareorganisation, eine Software einer vorgegebenen Größe zu einer bestimmten Fehlerrate zu produzieren
- Zeit – der Gesamtzeitplan des Projekts in Jahren
- Aufwand – der Gesamtaufwand für das Projekt in Personenjahren

Function-Points

Ein Function-Point ist eine Maßeinheit, mit der der Umfang der Geschäftsfunktionalität ausgedrückt wird, die ein Informationssystem für einen Benutzer bereitstellt. Die Kosten für einen einzelnen Function-Point werden auf Basis früherer Projekte berechnet.

Fünf „Standardfunktionen“ werden als Function-Points gezählt:

- Datenfunktionen:
  1. Interne Datenbestände (Internal Logical Files, ILF)
     - Tabellen in einer relationalen Datenbank
     - Anwendungssteuerungsinformationen

2. Externe Datenbestände (External Interface Files, EIF)
    - Tabellen in einer relationalen Datenbank; Anwendungssteuerungsinformationen, die nicht von der vorliegenden Anwendung gepflegt werden

- Transaktionsfunktionen:
    1. Eingabe (External Input, EI)
        - Dateneingabe der Benutzer
        - Daten- oder Dateifeeds von externen Anwendungen
    2. Ausgabe (External Output, EO)
        - Von der Anwendung erstellte Berichte einschließlich der abgeleiteten Daten
    3. Abfrage (External Inquiry, EQ)
        - Von der Anwendung erstellte Berichte werden gezählt, wobei der Bericht keine abgeleiteten Daten enthält

Identifizierte Funktionen werden anhand von drei Komplexitätsstufen gewichtet; die Anzahl der Punkte, die jeder Stufe zugewiesen werden, unterscheidet sich abhängig vom Typ des Function-Points.

Beispiel:

Komplexität	Punkte
Niedrig	7
Durchschnitt	10
Hoch	15

CoCoMo

Das CoCoMo (Constructive Cost Model) lässt den Aufwand und die Kosten für die Computersoftwareentwicklung als eine Funktion der Programmgröße zu. Die Größe der Software wird in EKLOC (Estimated Thousands of Lines of Code, geschätzte Tausend von Codezeilen) ausgedrückt.

CoCoMo wird auf drei Klassen von Softwareprojekten angewendet:

- Einfaches Projekt („Organic Mode“)
- Mittelschweres Projekt („Semi-Detached“)
- Komplexes Projekt („Embedded“)

Die Basis für CoCoMo-Gleichungen:

- **Aufwand (Effort Applied, E)** = $a_b(KLOC)^{b}{}_{b}$ **[Personenmonate]**
- **Entwicklungszeit (Development Time ,D)** = $c_b(Aufwand)^{d}{}_{b}$ **[Monate]**
- **Personen (People required, P)** = Aufwand / Entwicklungszeit **[Anzahl]**

KLOC – die geschätzte Anzahl der gelieferten Codezeilen (in Tausend) für das Projekt

a_b, b_b, c_b und d_b:

Softwareprojekte	a_b	b_b	c_b	d_b
Einfach	2.4	1.05	2.5	0.38
Mittelschwer	3.0	1.12	2.5	0.35
Komplex	3.6	1.20	2.5	0.32

7.5 Priorisierung (K2)	20 Minuten

Begriffe:

Skala, Dreistufenskala

7.5.1 Priorisierung (K2)

Die Priorisierung ermöglicht die Festlegung der relativen Wichtigkeit von Anforderungen, um die wichtigsten Anforderungen zuerst erfüllen zu können.

Die Priorisierung unterstützt die inkrementelle Entwicklung, da sie die Gruppierung von Anforderungen ermöglicht und die Festlegung der Prioritäten zur Implementierung.

7.5.2 Priorisierungsverfahren (K2)

Das Verfahren zur Festlegung der Prioritäten für Anforderungen umfasst die folgenden Aktivitäten:

1. Gruppierung von Anforderungen
   - Identifizieren von Anforderungen, die sich gegenseitig beeinflussen und die voneinander abhängen (Beispiel: Anforderungen, die eine komplexe Funktionalität erstellen)
2. Anforderungsanalyse
   - Analyse durch alle betroffenen Stakeholder, um die Wichtigkeitsstufe zu vereinbaren
   - Einflussanalyse als Mittel zur Unterstützung der Anforderungsanalyse
   - Festlegen der Prioritäten für Anforderungen
3. Erstellung des Anforderungsprojektplans
   - Erstellen eines Plans, bei dem Anforderungen mit hoher Priorität zuerst entwickelt werden sollen
   - Zuweisen von Verantwortung (wer implementiert die Anforderung)
4. Planen des inkrementellen Systemtests
   - Entwickeln von Testfällen zum Testen der einzelnen Systemschritte (erstellt auf Basis der priorisierten Anforderungen) basierend auf den Anforderungsprioritäten

7.5.3 Priorisierungsskala (K2)

Ein geläufiger Ansatz zur Priorisierung ist die Gruppierung von Anforderungen in Prioritätskategorien. Normalerweise wird die Dreistufenskala verwendet (Beispiel: Hoch, Mittel und Niedrig).

Da diese Skalen subjektiv und unpräzise sind, müssen alle betroffenen Stakeholder die Bedeutung der einzelnen Stufen in der verwendeten Skala vereinbaren. Die Priorität sollte klar definiert werden und sollte ein Schlüsselattribut für jede Anforderung sein.

Beispiele für Dreistufenskalen:

Bezeichnung	Beschreibung
Hoch	Eine wichtige Anforderung, die für den ersten Software-Release erforderlich ist.
Mittel	Unterstützt zwar notwendige Systemoperationen, könnte jedoch bis zu einem späteren Release warten, falls nötig.
Niedrig	Eine funktionale oder qualitative Verbesserung: "gewünscht, aber nicht notwendig".

Bezeichnung	Beschreibung
Essenziell	Das Produkt ist erst akzeptabel, wenn diese Anforderungen erfüllt sind.
Bedingt	Würde das Produkt verbessern, doch das Produkt ist nicht inakzeptabel, wenn dies fehlt.
Optional	Funktionen, die interessant sein könnten oder auch nicht.

7.6 Vereinbaren von Anforderungen (K2)	20 Minuten

Begriffe:

Freigabe

7.6.1 Vereinbarung (K2)

Die Vereinbarung von Anforderung, oft auch als Anforderungsfreigabe bezeichnet, ist eine formale Vereinbarung darüber, dass der Inhalt und Umfang der Anforderungen genau und vollständig ist.

Die formale Vereinbarung ist eine Grundlage des Projekts. Abstrakte Anforderungen (Geschäftsanforderungen) sollten vor dem Projektstart vereinbart werden. Detaillierte Anforderungen (funktionale und nicht funktionale Anforderungen) sollten vereinbart und freigegeben werden, bevor sie in die Implementationsphase übergehen.

Die Anforderungsfreigabe ist normalerweise die abschließende Aufgabe der Anforderungsanalyse und -entwicklung.

Die Anforderungsfreigabe sollte von den Stakeholdern des Projekts vorgenommen werden:

- Projektmanager sowohl auf Kundenseite als auch auf Anbieterseite
- Geschäftsbeauftragte des Kunden
- Geschäfts- und Systemanalysten
- Requirements Engineers
- Beauftragte der Qualitätssicherung und des Test- und Entwicklungsteams

Die Liste der Anforderungen muss für beide Seiten, also Kunde und Anbieter, bindend sein.

Einer der Gründe für die Anforderungsfreigabe besteht darin, sicherzustellen, dass die Anforderungen stabil sind und Änderungen über formale Änderungsanforderungen gesteuert werden müssen. Daher senkt die formale Vereinbarung das Risiko der Einführung neuer Anforderungen während oder gleich nach der Implementierung.

Die Vereinbarung von Anforderungen wird als vollständig erachtet, wenn alle relevanten Stakeholder des Projekts das Anforderungsdokument abgezeichnet (und damit freigegeben) haben.

Die Anforderungsfreigabe sollte dem Projektteam mitgeteilt werden und ist normalerweise ein Projektmeilenstein.

7.6.2 Vorteile der Anforderungsvereinbarung (K1)

- Durch die Vereinbarung wird sichergestellt, dass das richtige Produkt entwickelt wird (die Arbeiten basieren auf einer vereinbarten Liste von Anforderungen und behandeln nur das, was erforderlich ist, um einen Mehrwert für die Stakeholder zu schaffen).
- Minimiertes Risiko von Missverständnissen zwischen dem Kunden und dem Anbieter in Bezug auf den Projektumfang
- Grundlage für die weitere Entwicklungsarbeit

8 Verfolgen der Anforderungen (K2)	60 Minuten

Lernziele für das Foundation Level des Requirements Engineering
Die Lernziele legen fest, was Sie nach Beenden des jeweiligen Moduls gelernt haben sollten.

8.1 Rückverfolgung im Projekt (K2)

LO-8.1.1 Die Bedeutung der Rückverfolgbarkeit wiedergeben (K1)

LO-8.1.2 Typische Gründe für Änderungen bei den Anforderungen wiedergeben (K1)

LO-8.1.3 Den Zweck der Rückverfolgbarkeit beschreiben (K2)

LO-8.1.4 Die verschiedenen Arten von Rückverfolgbarkeit erkennen (K1)

8.2 Änderungsmanagement (K2)

LO-8.2.1 Die Eigenschaften des Änderungsmanagements beschreiben (K2)

LO-8.2.2 Die Struktur der Änderungssteuerungsgruppe wiedergeben (K1)

8.1 Rückverfolgung im Projekt (K2)	20 Minuten

Begriffe:

Horizontale und vertikale Rückverfolgung

8.1.1 Evolution der Anforderungen (K1)

Anforderungen sind nicht stabil, sondern entwickeln sich im Lauf des Projektlebenszyklus weiter.

Mögliche Gründe für eine Weiterentwicklung und vorgeschlagene Änderungen:

- Neue Erkenntnisse
- Neue Kundenbedürfnisse (resultierend aus neuen Vorschriften, Änderungen im Unternehmen, neuen Produkten etc.)
- Arbeitsfortschritt (d. h. nächste Projektphase, Verbesserung und Optimierung von bereits implementierten Funktionen etc.)
- Neue Verbindungen im Projekt (d. h. Integration in neue Systeme, neue Zugriffskanäle wie Internet oder mobile Kanäle etc.)

8.1.2 Rückverfolgbarkeit (K2)

Die Rückverfolgbarkeit stellt eine Lösung zur Steuerung von Weiterentwicklungen in den Anforderungen und anderen Artefakten, die sich auf diese Anforderungen beziehen, dar.

Die Rückverfolgbarkeit ist eine Möglichkeit zur Überprüfung, ob alle wichtigen Schritte im Entwicklungsprozess ausgeführt wurden. Sie sollte für alle Artefakte in beide Richtungen ausgeführt werden (Beispiel: Von den Anforderungen zu Design-Artefakten, und von Design-Artefakten zu den Anforderungen). Die Rückverfolgbarkeit ist auch wichtig für Tests, Verifizierung und Validierung.

Ziele der Rückverfolgbarkeit:

- Einflussanalyse
- Umfanganalyse
- Nachweis der Implementierung
- Verwendung der Anforderung (Anforderungsverfolgung als Nachweis, dass die Anforderungen angewendet werden und wie)

Um eine gute Rückverfolgbarkeit sicherzustellen, ist es wichtig, die Anforderungen eindeutig zu kennzeichnen.

8.1.3 Arten von Rückverfolgbarkeit (K2)

- Horizontale Rückverfolgung
  - Stellt die Abhängigkeiten zwischen Anforderungen derselben Ebene dar (Beziehungen zwischen verschiedenen Arten von Anforderungen).
- Vertikale Rückverfolgung
  - Stellt die Abhängigkeiten zwischen verschiedenen Artefakten dar (Anforderungen und Lösungs- und Leistungsspezifikation, Testfälle, Code, Module, Pläne etc.).

8.2 Änderungsmanagement (K2)	20 Minuten

Begriffe:

Änderung, Änderungssteuerungsgruppe, Änderungsmanagement, Änderungsanforderung

8.2.1 Änderungen an Anforderungen (K1)

Änderungen an Anforderungen können zu jedem Zeitpunkt der Realisierung des Projekts und nach der Freigabe der finalen Software in der Produktionsumgebung angefordert werden. Änderungen kommen immer vor und es ist wichtig, Änderungen entsprechend des Prozesses und zeitlich zu planen.

Mögliche Ursachen für Änderungen:

- Erweiterung der vorhandenen Funktionalität
- In der Software/Dokumentation gefundene Fehler
- Andere Dinge, die bei Tests gefunden wurden, wie zum Beispiel schlechte Leistung etc.
- Anfrage für eine neue Funktionalität oder Modifikation einer vorhandenen Funktionalität
- Änderungen als Folge externer Faktoren (organisatorische Änderungen, Änderungen an Vorschriften)

8.2.2 Änderungsmanagement (K2)

Der Prozess des Änderungsmanagements ist der Vorgang, Änderungen an einem Softwaresystem, an Dokumenten oder anderen Produkten des Projekts zu planen, zu implementieren und zu evaluieren. Der Zweck des Änderungsmanagements besteht darin, die Verarbeitung von Änderungen zu unterstützen und die Rückverfolgbarkeit von Änderungen sicherzustellen.

Der Prozess des Änderungsmanagements enthält die folgenden Aktivitäten:

1. Identifizierung potenzieller Änderungen
2. Anforderung einer neuen Funktionalität
3. Analyse der Änderungsanforderung

4. Bewerten der Änderung
5. Planen der Änderung
6. Implementieren der Änderung
7. Überprüfen und Abschluss der Änderung
8. Rollout der Änderung

Abhängig von der Komplexität und den Auswirkungen kann eine Änderung verschiedene Einflüsse auf das System ausüben. Kleine Änderungen können geringfügige Modifikationen erfordern, während komplexe Änderungen die Logik des Systems radikal ändern können. Jede Änderung sollte sorgfältig analysiert werden, um die damit verbundenen Risiken zu erkennen und den Wert der Modifikation gegen die vorhergesagten Risiken abwägen zu können.

8.2.3 Änderungsanforderung (K2)

Eine Änderung sollte als formales Änderungsanforderungsdokument eingereicht werden (auch als „Request For Change" (RFC) bezeichnet). Dieses Dokument beschreibt den Grund für die Änderung, die Priorität und die angeforderte Lösung zusammen mit weiteren Details wie:

- Name der Person/Abteilung oder anderen Entität, die die Änderung anfordert
- Datum der Einreichung
- Geplantes Datum der Implementierung der Änderung (falls zutreffend)
- Kosten für die Änderung

8.2.4 Änderungssteuerungsgruppe (K1)

Änderungen werden von einer Änderungssteuerungsgruppe (Change Control Board, CCB) geprüft und festgelegt. Durch die CCB wird die Implementierung der Änderung oder eine vorgeschlagene Änderung an einem Produkt oder Service kontrolliert gesteuert.

Die CCB ist ein Ausschuss, der basierend auf Informationen (wie das Risiko einer Änderung, deren Auswirkungen, der erforderliche Implementierungsaufwand) Entscheidungen darüber trifft, ob die vorgeschlagenen Änderungen implementiert werden sollen. Die CCB setzt sich aus Projekt-Stakeholdern oder deren Vertreter zusammen.

Die CCB kann die folgenden Rollen umfassen:

- Projektmanagement
- Entwicklungsmanagement

- Requirements Engineers
- Qualitätssicherung (Qualitätsmanagement, Testmanagement)
- Unternehmensmanagement, falls zutreffend
- Kunde, falls zutreffend

8.2.5 Lebenszyklus einer Anforderung (K2)

Abhängig vom Grad der Analyse und/oder Implementierung der Anforderung wird der Anforderung jeweils ein anderer Status zugewiesen. Der Lebenszyklus einer Anforderung kann beispielsweise durch die folgenden Statusangaben ausgedrückt werden:

- Neu (vorgeschlagen)
- Zur Überprüfung
- Genehmigt
- Konflikt
- Implementiert
- Modifiziert
- Gelöscht
- Getestet
- Bereitgestellt

Verschiedene Ansätze und Organisationen verwenden möglicherweise einen anderen Anforderungslebenszyklus (und damit andere Statusangaben). In vielen Fällen sind die Lebenszyklen der Anforderungen, die Änderungsanforderungen und Fehler sehr ähnlich und werden mit demselben Werkzeug gesteuert.

8.2.6 Unterscheidung zwischen Fehlermanagement und Änderungsmanagement (K2)

Es ist wichtig, zwischen einer Änderung und einem Fehler zu unterscheiden. Ein Fehler wird als Makel in einer Komponente oder in einem System definiert, der dazu führen kann, dass die Komponente oder das System die erforderliche Funktion nicht ausführen kann. Es handelt sich dabei um eine Abweichung vom erforderlichen Zustand des Systems. Eine Änderung dagegen ist eine Modifikation der vorhandenen Features, Anforderungen oder Funktionen oder fügt neue hinzu.

8.2.7 Auswirkungen einer Änderung auf das Projekt (K2)

Änderungen an Anforderungen können verschiedene Auswirkungen auf das Projekt haben. Die häufigsten Auswirkungen:

- Änderung des Zeitplans, Budgets oder der Ressourcen
- Arbeiten, die sich durch die Änderung ergeben (abhängig von der Projektphase):
    - Aktualisieren der Analyse und der Design-Artefakte (Beispiel: Spezifikationen)
    - Aktualisieren der technischen Dokumentation und Benutzerdokumentation
    - Änderungen an der Teststrategie und an den Tests
    - Aktualisieren des Testplans
    - Aktualisieren des Schulungsbedarfs/-plans
    - Erweitern/Kürzen des Programmierungsaufwands
    - Ändern der Testvorbereitungen und des Ausführungsumfangs

9 Qualitätssicherung (K2)	30 Minuten

Lernziele für das Foundation Level des Requirements Engineering
Die Lernziele legen fest, was Sie nach Beenden des jeweiligen Moduls gelernt haben sollten.

9.1 Einflussfaktoren (K1)

LO-9.1.1 Die Faktoren wiedergeben, die Einfluss auf das Requirements Engineering haben (K1)

9.3 Qualitätssicherung durch Testbarkeit (K2)

LO-9.3.1 Erklären, wie Produkte des Requirements Engineering den Testvorgang unterstützen (K2)

LO-9.3.2 Die Verwendung der Abnahmekriterien beschreiben (K2)

LO-9.3.3 Beschreiben, wie das Requirements Engineering zum Testvorgang beitragen kann (K2)

9.4 Metriken (K2)

LO-9.4.1 Die Definition von Metrik wiedergeben (K1)

LO-9.4.2 Beschreiben, welche Metriken für das Requirements Engineering verwendet werden können (K2)

9.1 Einflussfaktoren (K1)	10 Minuten

9.1.1 Einflüsse auf das Requirements Engineering (K1)

Die Qualität der Produkte des Requirements Engineerings hängt von den folgenden Faktoren ab:

- Das Produkt, das hergestellt wird
- Die Umgebung, in der das Produkt hergestellt wird
- Die Domäne (Komplexität der Geschäftsdomäne, die Innovationsstufe, die Häufigkeit von Änderungen im Unternehmen etc.)
- Rechtlich Faktoren, Sicherheits- und Umweltfaktoren
- Zeit- und Kostendruck (Zeit- und Kostenbeschränkungen können die Fähigkeit zur Durchführung von Requirements Engineering-Prozessen vermindern)
- Kulturelle Faktoren (Sprache, Ausbildung etc.)
- Technologie- und Designbeschränkungen

Diese Faktoren müssen bei der Planung der Qualitätssicherungsaktivitäten berücksichtigt werden.

9.2 Verifizierung der Anforderung in der Anforderungserhebungsphase (K2)	20 Minuten

Eine Verifizierung muss während der Entwicklung einer Lösung ständig vorgenommen werden. Zur Verifizierung von Anforderungen in den Erhebungsphasen können einige der folgenden Methoden angewendet werden:

- Überprüfungsmethoden
- Simulationen
- Präsentationen
- Demonstrationen
- Demos

Es ist wichtig, die Verifizierung ab Projektbeginn einzuplanen.

9.3 Qualitätssicherung durch Testbarkeit (K2)	20 Minuten

Begriffe:

Abnahmekriterien, Testbarkeit

9.3.1 Requirements Engineering und Testvorgang (K2)

Das Requirements Engineering ist eng mit dem Testvorgang verbunden. Gute Testfälle erfordern gute Anforderungen, die getestet werden können. Die Einbeziehung von Testern für die Spezifikation ist daher sehr wichtig.

9.3.2 Abnahmekriterien (K2)

Gemäß der Fachbezeichnung PRINCE2 sind Abnahmekriterien (auch Erfolgskriterien genannt) die Standards, die zur Erfüllung der Qualitätserwartungen des Kunden und zur Erreichung der Abnahme des finalen Produkts durch den Kunden erforderlich sind. Mit anderen Worten sind es Kriterien, die für die angegebene Funktion, Komponente oder jedes andere Element des Softwareprodukts oder andere Ausgabe des Projekts festgelegt wurden, die erfüllt werden müssen, um den Kunden zufrieden zu stellen und zu erreichen, dass er das Projekt abnimmt.

Die Abnahmekriterien sollten durch beide Seiten (Kunde und Anbieter) vereinbart werden, bevor das Projekt gestartet wird (sie sollten ein Teil der Vertragsdokumentation sein) und bilden die Grundlage für den Projektqualitätsplan. Jede abstrakte Anforderung muss mindestens ein Abnahmekriterium aufweisen. Diese Kriterien sind die Grundlage für die Abnahmetests.

Alle Kriterien müssen messbar und die Messmethode für die Kriterien muss realistisch und vereinbart sein.

Beispiel eines Abnahmekriteriums: „Anwendung muss in weniger als einer Sekunde reagieren oder eine Wartemeldung anzeigen“

9.3.3 Testmethoden (K2)

Die Produkte des Requirements Engineering unterstützen den Testvorgang durch Bereitstellung einer sogenannten Testbasis. Anforderungen und deren Spezifikationen können eine Testbasis sein.

Die Produkte des Requirements Engineering können den Testvorgang unterstützen durch:

- Ermöglichen der funktionalen Abdeckung (Abdecken aller funktionalen Anforderungen durch Testfälle; Testfälle werden geschrieben, um alle funktionalen Anforderungen abzudecken)

  Funktionale Abdeckung = ∑ der Anforderungen, die anhand von Testfällen getestet wurden / ∑ der Anforderungen
- Bereitstellen einer zuverlässigen Testbasis für Black-Box-Tests oder spezifikationsbasierte Tests wie:
  - Grenzwerte für Dateneintragskategorien
  - Anwendungszustände
  - Logische und geschäftliche Beschränkungen etc.
- Ermöglichen von Testmethoden wie: Äquivalenzklassenbildung, Grenzwertanalyse, Entscheidungstabellentest, zustandsbasierter Test, anwendungsfallbasierter Test

Beispiel: Äquivalenzklassenbildung – ist eine Testmethode, die die in einem bestimmten Modul einer Software verwendeten Eingabedaten in Datenklassen aufteilt, aus denen Testfälle abgeleitet werden können. Testfälle sind darauf ausgelegt, jede Klasse mindestens einmal abzudecken.

Beispiel einer Äquivalenzklassenbildung:

1. Der gültige Bereich für das Alter des Benutzers ist 1 bis 99. Der gültige Bereich wird als gültige Klasse bezeichnet. Es gibt zwei weitere Klassen ungültiger Bereiche. Die erste ungültige Klasse wäre ≤ 0 und die zweite ungültige Klasse wäre ≥ 100.
2. Die gültige Klasse für Antworten in einem Fragebogen enthält die folgenden Textzeichen: A, B, C, D. Die ungültige Klasse enthält alle anderen Textzeichen, einschließlich Sonderzeichen.

Testfälle sollten die gültige Klasse und beide ungültigen Klassen berücksichtigen.

Es können auch andere Testmethoden verwendet werden: Grenzwertanalyse, Entscheidungstabellentest, zustandsbasierter Test, Testfalltest etc. Sie sollten nach einer Testanalyse der Anforderungen eingesetzt und entsprechend des Anforderungsinhalts angewendet werden.

9.3.4 Anforderungen und Testprozess (K2)

Anforderungen sind die grundlegenden Eingabeinformationen für den Prozess der Entwicklung und zum Testen des Systems. Gut definierte Anforderungen vermindern das Risiko des Projektausfalls (oder sogar Produktausfalls), da sie sorgfältige Tests ermöglichen. Die Stabilität der Anforderungen ermöglicht es, Termine für die spezifische Aufgabe einzuhalten.

Es ist wichtig zu wissen, dass die Anforderungen anhand von statischen Tests validiert werden sollten (wozu Tester erforderlich sein können) und von Testmanagern abgenommen werden sollten (weil sich die Anforderungen in einigen Fällen als nicht testbar herausstellen können). Tester können dazu beitragen, die Qualität der Anforderungen zu verbessern, weil sie auf Schwachpunkte und mögliche Fehler hinweisen. Tester sollten auch an der Überprüfung von Anforderungen beteiligt sein, um deren Testbarkeit sicherzustellen.

9.4 Metriken (K2)	20 Minuten

Begriffe:

Metrik

9.4.1 Metrik (K1)

Metrik bezeichnet eine Maßeinheit und die Methode, die zur Messung verwendet wird [ISO 14598].

Metriken ermöglichen eine quantifizierbare Aussage zum Projektstatus und der Qualität.

Es ist wichtig, zu wissen, dass die Messergebnisse (bei der Messung gesammelten Zahlen) immer mit den Referenzdaten (relevante Metrik) verglichen werden müssen.

9.4.2 Metriken für Anforderungen (K1)

Die folgenden Metriken können auf Anforderungen angewendet werden:

- Projektkosten
  - Anzahl der Anforderungen
- Projektverfolgung
  - Anzahl der Anforderungen, die auf andere Artefakte rückverfolgt werden
- Projektstabilität
  - Anzahl der änderbaren Anforderungen
- Prozessverbesserung
  - Gründe für Änderungen an Anforderungen (Fehler, Verbesserungen)

- Qualität der Spezifikation
  - Abdeckung der Anforderungen durch Modelle
- Anzahl der Fehler
  - Fehlertyp
  - Anzahl der Fehler in Anforderungen mit verschiedenen Typen (Logik, Konsistenz, Datenfehler etc.)

9.4.3 Messen der Anforderungsqualität (K2)

Einige Fragen ermöglichen die Evaluierung der Qualität der Anforderungen:

- Sind die Anforderungen korrekt?
- Sind die Anforderungen verständlich?
- Sind die Anforderungen realisierbar?
- Sind die Anforderungen rückverfolgbar?
- Sind die Anforderungen identifizierbar?
- Sind die Anforderungen testbar?

Einige Formeln zum Messen der Anforderungsqualität:

> Klarheit = ∑ der Anforderungen ohne gemeldete Fehler und Probleme / ∑ der Anforderungen

Die Änderungsrate kann auch ein Maß für die Qualität der Anforderungen sein (dies trifft nicht auf agile Projekte zu). Je höher die Änderungsrate bei allen Anforderungen ist (was durch Anstrengungen verursacht werden kann, die zur Klärung nicht verständlicher Anforderungen und/oder einer inkonsistenten Beschreibung der Anforderungen führen), desto mehr ist das Projekt gefährdet. Daher sollte die Änderungsrate gemessen werden, um die Risiken innerhalb eines Projekts zu steuern.

10 Werkzeuge (K2)	40 Minuten

Lernziele für das Foundation Level des Requirements Engineering
Die Lernziele legen fest, was Sie nach Beenden des jeweiligen Moduls gelernt haben sollten.

10.1 Vorteile von Werkzeugen (K2)

LO-10.1.1 Den Zweck der Werkzeugunterstützung für das Requirements Engineering erklären (K2)

LO-10.1.2 Beschreiben, welche Aktivitäten durch Werkzeuge im Requirements Engineering unterstützt werden können (K2)

10.2 Werkzeugkategorien (K2)

LO-10.2.1 Welche Anforderungen gelten für Werkzeuge im Bereich des Requirements Engineering? (K2)

LO-10.2.2 Was muss beachtet werden bezüglich der Kosten für Werkzeuge? (K2)

10.1 Vorteile von Werkzeugen (K2)	20 Minuten

Begriffe:

Werkzeuge des Requirements Engineering

10.1.1 Anwendungen für Werkzeuge des Requirements Engineering (K2)

Werkzeuge zum Speichern und Verwalten von Anforderungen erleichtern das Requirements Engineering. Sie können wiederholte mechanische Aktivitäten sein oder einen Überblick sicherstellen. Daher ist es möglich, schwierige Dokumente konsistent und aktuell zu halten. Die Auswahl eines Werkzeugs muss vor der Entwicklung eines Produkts erfolgen. Andernfalls können bestimmte Situationen zu erheblichen Problemen führen.

Werkzeuge können die folgenden Aktivitäten im Requirements Engineering unterstützen:

- Identifizierung und Speicherung von Anforderungen
- Anforderungsmodellierung (einschließlich Prototyping)
- Anforderungsdokumentation (Erstellen von Anforderungsspezifikationen)
- Definieren und Pflegen der Rückverfolgbarkeit von Anforderungen

10.1.2 Vorteile des Einsatzes von Werkzeugen (K2)

- Sicherstellen, dass alle Anforderungen am selben Ort gespeichert werden und für alle einbezogenen Stakeholder zugänglich sind
- Unterstützen der Rückverfolgbarkeit von Anforderungen (Testfälle etc.) und Ermöglichen der Verifizierung der relevanten Anforderungsabdeckung
- Ermöglichen der einfachen Steuerung von Anforderungsänderungen
- Verbessern der Qualität der Anforderungsspezifikation durch Erzwingen der Verwendung definierter Dokumentenvorlagen und Modellierungssprachen
- Zeitersparnis durch Automatisieren einiger Aktivitäten (wie Generieren vollständiger Spezifikationen aus dem Werkzeug)

10.2 Werkzeugkategorien (K2)	20 Minuten

Begriffe:

Werkzeugkategorien

10.2.1 Werkzeugkategorien (K2)

- Werkzeuge zur Anforderungserhebung
  - Mindmapping
- Modellierungswerkzeuge
  - UML-Werkzeuge
  - SysML-Werkzeuge
- Prototyping-Werkzeuge
- Werkzeuge für das Anforderungsmanagement
- Werkzeuge für das Fehlermanagement
- Werkzeuge für das Änderungsmanagement
- Werkzeuge für das Projektmanagement

Die Kosten für die Beschaffung der Werkzeuge variieren sehr stark. Im Handel erhältliche Werkzeuge können sehr teuer sein, während Open-Source-Werkzeuge kostenlos sind. Ein Werkzeug muss daher sehr sorgfältig gewählt werden. Vor der Auswahl eines Werkzeugs sollte eine Analyse durchgeführt werden. Die Analyse sollte Folgendes berücksichtigen:

- Welche Modellierungssprache wird in der Organisation verwendet und sollte durch das Werkzeug unterstützt werden?
- Plant die Organisation, zukünftig andere Modellierungssprachen zu verwenden? (In diesem Fall kann es vernünftig sein, ein Werkzeug zu kaufen, das die geplante Sprache und gleichzeitig die in der Organisation aktuell verwendete Sprache unterstützt.)
- Die Anforderungen einer Organisation hinsichtlich der Funktionalität, die vom Werkzeug erwartet wird (normalerweise bieten im Handel erhältliche Werkzeuge komplexere Funktionen als Open-Source-Werkzeuge)
- Die Kosten für das Werkzeug, wobei berücksichtigt werden sollte, ob das Werkzeug nur für ein spezifisches Projekt verwendet werden soll oder für alle bzw. die meisten Projekte

- Ob das Werkzeug in andere notwendige Werkzeuge integriert werden kann (wie zum Beispiel Werkzeuge für das Fehlermanagement und das Projektmanagement und andere Werkzeuge, abhängig von den Bedürfnissen der Organisation)
- Ob das Werkzeug Informationen mit einem Werkzeug der Kundenorganisation austauschen kann (in einigen Fällen führt der Kunde eigene Anforderungsanalysen durch; in diesem Fall sollten die Produkte der detaillierten Anforderungsanalyse des Anbieters in die Kundenumgebung migriert werden können)
- Benutzerfreundlichkeit und leichte Erlernbarkeit (potenzielle Schulungskosten), Verfügbarkeit von Online-Hilfe, Handbüchern, Lernprogrammen, weiterer Support

Eine hastig getroffene Wahl kann hohe Kosten verursachen (K2):

- Kosten für die Beschaffung eines Werkzeugs, die die Benutzeranforderungen sowie seinen Zweck nicht erfüllen
- Kosten für die Beschaffung eines teuren Werkzeugs, das nur für ein Projekt verwendet wird, oder Kauf eines teueren Werkzeugs, obwohl Open-Source-Werkzeuge mit ähnlicher Funktionalität verfügbar sind
- Schulungskosten für den Fall, dass ein Werkzeug gekauft wird, das kein ausreichendes Hilfesystem bietet (besonders in dem Fall, wenn nur Basisfunktionen des Werkzeugs benötigt werden)
- Kosten der Erweiterung des gekauften Werkzeugs durch zusätzliche Funktionen, die die Benutzer benötigen, aber nicht vom Werkzeug unterstützt werden (obwohl es andere Werkzeuge mit dieser Funktionalität gäbe)
- Kosten der Integration des Werkzeugs in andere in der Organisation verwendete Werkzeuge

11 Literaturangabe

Beck, K.: *Extreme Programming*. Munich 2003

Beck, K.: *Extreme Programming Explained: Embrace Change*. Boston 2000

Beck, K.: *Test Driven Development. By Example*. Amsterdam 2002

Beck, K.: *Refactoring*: *Improving the Design of Existing Code*. Addison-Wesley Longman 1999

Boehm, B.: *Software Engineering Economics*. Englewoods Cliffs, NJ 1981

Bohner, S.A. and R.S. Arnold, Eds. *Software Change Impact Analysis*. Los Alamitos, California, USA, IEEE Computer Society Press 1996.

Bundschuh, M.; Fabry, A.: *Aufwandschätzung von IT-Projekten*. Bonn 2004

Cockburn, A.: Agile *Software Development*. Addison Wesley 2002

Cockburn, A.: *Writing Effective Use Cases*. Amsterdam 2000

Cohn M.: *Estimating With Use Case Points*, Fall 2005 issue of Methods & Tools

Cotterell, M. and Hughes, B.: *Software Project Management*, International Thomson Publishing 1995

Newman, W.M. and Lamming, M.G.: *Interactive System Design*, Harlow: Addison-Wesley 1995

Davis A. M.: *Operational Prototyping: A new Development Approach*. IEEE Software, September 1992. Page 71

Davis, A. M.: Just *Enough Requirements Management. Where Software Development Meets Marketing*, Dorset House, 2005, ISBN 0932633641

DeMarco, T. et al.: *Adrenalin-Junkies und Formular-Zombies – Typisches Verhalten in Projekten*. Munich 2007

DeMarco, T.: *Controlling Software Projects: Management, Measurement and Estimates*. Prentice Hall 1986

DeMarco, Tom: *The Deadline: A Novel About Project Management*. New York 1997

Dorfman, M. S.: *Introduction to Risk Management and Insurance (9 ed.)*. Englewood Cliffs, N.J: Prentice Hall 2007. ISBN 0-13-224227-3.

Dynamic Systems Development Method Consortium. See: http://na.dsdm.org

Ebert, Ch.: *Systematisches Requirements Management. Anforderungen ermitteln, spezifizieren, analysieren und verfolgen*. Heidelberg 2005

Evans, E. J.: *Domain-Driven Design*: *Tackling Complexity in the Heart of Software*. Amsterdam 2003

Graham, D. et al: *Foundations of Software Testing*. London 2007

Gilb, T.; Graham, D.: *Software Inspection*. Reading, MA 1993

Gilb, T.: *What's Wrong with Requirements Specification.* See: www.gilb.com

Heumann, J.: *The Five Levels of Requirements Management Maturity,* see: http://www.ibm.com/developerworks/rational/library/content/RationalEdge/feb03/ManagementMaturity_TheRationalEdge_Feb2003.pdf

Linstone H. A., Turoff M.: *The Delphi Method: Techniques and Applications, Reading*, Mass.: Adison-Wesley, ISBN 9780201042948, 1975

Hull, E. et. All: *Requirements Engineering*. Oxford 2005

IEEE Standard 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology

IEEE Standard 829-1998 IEEE Standard for Software Test Documentation

IEEE Standard 830-1998 IEEE Recommended Practice for Software Requirements Specifications

IEEE Standard 1012-2004: IEEE Standard for Software Verification and Validation

IEEE Standard 1059-1993: IEEE guide for software verification and validation plans

IEEE Standard 1220-1998: IEEE Standard for Application and Management of Systems Engineering Process

IEEE Standard 1233-1998 IEEE Guide for Developing System Requirements Specifications

IEEE Standard 1362-1998 IEEE Guide for Information Technology-System Definition – Concept of Operations (ConOps) Document

ISO 9000

ISO/EIC 25000

ISO 12207

ISO 15288

ISO 15504

ISO 31000: RiskManagement - Principles and Guidelines on Implementation

IEC 31010: Risk Management - Risk Assessment Techniques

ISO/IEC 73: Risk Management – Vocabulary

ISTQB: ISTQB Glossary of Testing Terms 2 1

ISTQB: Certified Tester, Foundation Level Syllabus, version 2011

Jacobsen, I. et al.: *The Unified Software Development Process*. Reading 1999

Jacobson, I.et al.: *Object-Oriented Software Engineering. A Use Case Driven Approach*. Addison-Wesley 1993

Kilpinen, M.S.: *The Emergence of Change at the Systems Engineering and Software Design Interface: An Investigation of Impact Analysis. PhD Thesis*. University of Cambridge. Cambridge, UK 2008.

Lauesen, S.: *Software Requirements: Styles and Techniques*. London 2002

Mangold, P.: IT-*Projektmanagement kompakt*. Munich 2004

McConnell, S.: *Aufwandschätzung für Softwareprojekte*. Unterschleißheim 2006

McConnell, S.: *Rapid Development: Taming Wild Software Schedules (1st ed.)*. Redmond, WA: Microsoft Press. ISBN 1-55615-900-5, 1996

Paulk, M., et al: *The Capability Maturity Model: Guidelines for Improving the Software Process*. Reading, MA 1995

Pfleeger, S. L.: *Software Engineering: Theory and Practice, 2nd edition*. Englewood Cliffs, NJ 2001

Pfleeger, S.L. and J.M. Atlee: *Software Engineering: Theory and Practice.* Upper Saddle River, New Jersey, USA, Prentice Hall 2006.

Pohl, K.: *Requirements Engineering. Grundlagen, Prinzipien, Techniken*. Heidelberg 2007

Project Management Institute: *A Guide to the Project Management Body of Knowledge* (PMBOK ® Guide). PMI 2004

Putnam, L.e H.; Ware M. *Measures for excellence: reliable software On time, within budget*. Yourdon Press. ISBN 0-135676-94-0, 1991

Robertson, S.; Robertson, J.: *Mastering the Requirements Process*, Harlow 1999

Rupp, C.: *Requirements-Engineering und Management. Professionelle, Iterative Anforderungsanalyse in der Praxis*. Munich 2007

Sharp H., Finkelstein A. and Galal G.: *Stakeholder Identification in the Requirements Engineering Process*, 1999

Sommerville, I.: *Requirements Engineering*. West Sussex 2004

Sommerville, I.: *Software Engineering 8*. Harlow 2007

Sommerville, I.; Sawyer, P.: *Requirements Engineering: A Good Practice Guide*. Chichester 1997

Sommerville, I.; Kotonya, G.: *Requirements Engineering: Processes and Techniques*. Chichester 1998

Spillner, A. et all: *Software Testing Foundations*. Santa Barbara, CA 2007

SWEBOK - The Guide to the Software Engineering Body of Knowledge: http://www.computer.org/portal/web/swebok/home

Thayer, R. H.; Dorfman, M.: *Software Requirements Engineering, 2nd edition*. Los Alamitos, CA 1997

V-Modell® XT: http://www.vmodellxt.de/

Wiegers, K.E.: *First Things First: Prioritizing Requirements.* Software Development, September 1999

Wiegers, K. E.: *Software Requirements*. Redmond 2005

Wiegers, K. E.: *More About Software Requirements: Thorny Issues and Practical Advice*. Redmond, Washington 2006

Young, R. R.: *Effective Requirements Practices*. Addison-Wesley 2001

12 Index