

Abschnitt 1.10.2.4

DOKUMENTENORIENTIERTE TECHNIKEN

09.11.2018

Software Engineering 2

117

Systemarchäologie

- Informationen aus der Dokumentation oder der Implementierung eines Alt- oder Konkurrenzsystems gewinnen
- Interessant, wenn das Wissen über die implementierte Fachlogik verloren ist
- Erzeugt viele, sehr detaillierte Anforderungen, ist aufwendig
- Einzige Möglichkeit zum vollständigen Transfer der Funktionalität von alt nach neu

Perspektivenbasiertes Lesen

- Text wird aus einer spezifischen Perspektive (z.B. Tester oder Realisierer) gelesen
- Für die Perspektive irrelevante Inhalte werden ignoriert
- Auf speziellen Bereich fokussierte Analyse bestehender Dokumente

Systemarchäologie: Analyse des Codes stellt sicher, dass keine bereits implementierte Funktionalität vergessen wird und mindestens die Fachlogik des Altsystems erhoben wird.

Falls es deutliche Abweichungen beim Funktionsumfang zwischen altem und neuem System gibt, so empfiehlt sich frühzeitig der parallele Einsatz einer anderen Ermittlungstechnik, z.B. einer Kreativitätstechnik

Perspektivenbasiertes Lesen ermöglicht es z.B. detaillierte, technologie- und umsetzungsabhängige Inhalte von essentiellen, fachlichen Inhalten, die für ein Nachfolgesystem in Betracht kommen, zu trennen. Kann auch zur Prüfung von Anforderungsdokumenten eingesetzt werden, indem jeweils nur auf einen Qualitätsaspekt geschaut wird.

Wiederverwendung

- Einmal erhobene und auf entsprechendem qualitativen Stand dokumentierte Anforderungen können wieder verwendet werden
- Ideal ist die Speicherung der Anforderungen in einer Datenbank mit entsprechenden Sichten und Suchmöglichkeiten

Wiederverwendung kann die Kosten für die Anforderungsermittlung drastisch senken

Abschnitt 1.10.2.5

BEOBACHTUNGSTECHNIKEN

09.11.2018

Software Engineering 2

120

- Gut in Situationen, in denen die Fachleute ihr Wissen nicht formulieren können oder keine Zeit haben, dieses an den Requirements Engineer weiterzugeben
- Beobachten der Stakeholder bei ihrer Arbeit, Arbeitsschritte dokumentieren
- Daraus zu unterstützende Arbeitsschritte, mögliche Fehler, Risiken und Fragen ermitteln
- Stakeholder können ihr Wissen aktiv demonstrieren oder nur beobachtet werden
- Beobachtetes hinterfragen! Geht es besser?
- Externer Beobachter hat bessere Chancen Ineffizienz zu erkennen

„Nobody can talk better about what they do and why they do it than they can while in the middle of doing it.“
— Beyer, Holtzblatt: Contextual Design:
Defining Customer-Centered Systems

Beobachtete Arbeitsschritte, Fehler, Probleme,... sind alles potenzielle Kandidaten um daraus Anforderungen zu extrahieren

Wenn man das Beobachtete nicht hinterfragt besteht die Gefahr, veraltete Technologieentscheidungen und ineffiziente Prozesse (Ist-Situation) zur Anforderung für das neue System zu machen. Ein externer Beobachter hat es hier leichter, da er nicht in dem gelebten Prozess sozialisiert ist, daher kann er leichter kritisch reflektieren.

Beobachtungstechniken eignen sich gut, um detaillierte Anforderungen und Basismerkmale zu ermitteln. Der Requirements Engineer kann Basismerkmale erkennen, die der Stakeholder gar nicht mehr (bewusst) wahrnimmt. Weiterer Vorteil ist, dass der Requirements Engineer den Fachjargon der Anwendungsdomäne kennen lernt.

Begeisterungs-/Leistungsfaktoren können nur gefunden werden, wenn sie bereits im Prozess integriert sind. Eignet sich dementsprechend nicht für völlig neue Prozesse.

Feldbeobachtung

- Requirements Engineer ist vor Ort bei den Fachexperten und beobachtet / dokumentiert ihre Arbeitsprozesse, Handgriffe, Arbeitsabläufe
- Video- oder Audio-Aufnahmen können unterstützen
- Gut bei schwer sprachlich zu vermittelnden Prozessen, die Essenz des Prozesses muss aber beobachtbar sein

Apprenticing

- Idee: „In die Lehre gehen“
- Requirements Engineer übernimmt Aufgaben des Stakeholder
- Requirements Engineer als Lehrling muss alles unklare sofort hinterfragen
- So lernt er alle Anforderungen aus erster Hand kennen und erfährt insbesondere Anforderungen, die für den Stakeholder selbstverständlich sind

Apprenticing bietet auch einen psychologischen Vorteil: Das übliche Machtverhältnis zwischen Requirements Engineer und Fachspezialist wird umgekehrt. Stakeholder ist in der Rolle des Meisters, der über Wissen verfügt, das dem Lehrling noch fehlt.

Abschnitt 1.10.2.6

ERGÄNZENDE TECHNIKEN

Unterstützende Techniken stammen aus der Moderation, Kommunikationspsychologie, Kognitionswissenschaft aber auch der Softwaretechnik

- **Personas:** Eine erfundene Persönlichkeit, die als Stellvertreter einer nicht verfügbaren Klasse von Stakeholdern dient
- **Workshops:** Viele Stakeholder beisammen, intensive Auseinandersetzung, oftmals für Anforderungsgewinnung der Benutzerschnittstelle. Besonders relevant: Business Use-Case Workshops

Personas sind gerade bei Entwicklungen für den Markt wichtig, da in diesem Fall die Benutzer typischerweise nicht greifbar sind. Sie werden durch archetypische Personas vertreten. Die Persona muss detailliert ausgearbeitet sein, um eine echte Hilfe bei der Beurteilung ihrer Anforderungen zu sein. Das kann z.B. so aussehen:

Name: Anne

34 Jahre alt

Seit 3 Jahren Mitarbeiterin der IT-Abteilung der Credit Suisse als Business Analyst
liest viel, ist an ihrem Job sehr interessiert, möchte sich dort ständig verbessern,
hört Musik und kocht gerne

Mag keine Leute, die keine prägnanten Aussagen zustande bringen

Mag IT-Gadgets

Hat eine Katze

Am besten mit einem Bild arbeiten, welches die Persona darstellt.

Präzision ist bei der Entwicklung einer Persona wichtig.

- **Persona = prototypischer Benutzer.** Modelliert deren Ziele, Verhaltensweisen und Eigenschaften, die im Hinblick auf das zu entwickelnde System relevant sind.
- **Ziele**
 - Identifikation typischer Benutzergruppen
 - Genaues Verständnis ihres Verhaltens und ihrer Eigenschaften, um so ihre spezifischen Bedürfnisse besser zu verstehen
 - Empathie des Entwicklungsteams für Persona als Prototyp der Benutzergruppe: „Wir bauen das System für Paul Professor und Stefan Student“
- **Warum?**
 - System, das allen Benutzergruppen gleichermaßen gerecht wird kaum möglich: z.B. Senior (85) vs. Teenager (16)
 - Benutzergruppen haben verschiedene Bedürfnisse: z.B. Anfänger vs. (ewiger) Fortgeschrittener

Personas lassen sich etwa aus Umfragen extrahieren, indem Personen mit ähnlichen Antworten zu einer Persona abstrahiert werden.

Zur Vorbereitung der Umfrage benötigt man allerdings schon eine Hypothese über die Einflussfaktoren

Personas – Beispiel



- **Name:** Paul Professor
- **Alter und biographische Angaben:** 43, ...
- **Ziele:** Stundenplanung für seine Fakultät, zufriedene Kollegen
- **Aufgaben:** Stundenplan in zwei Tagen fertigstellen
- **Kenntnisse:** Seit 8 Jahren Informatik-Professor, fortgeschrittene Kenntnisse in Excel, Programmiererfahrung in Scala, Java, Ruby,..., Experte für formale Programmanalyse, Erfahrung mit OO-Design und Softwarearchitektur
- **Einstellungen:** Beschäftigt sich gerne mit Software, mag IT-Gadgets, kocht gerne italienisch, betreibt Gesellschaftstanz, hört gerne Verdi-Opern
- **Erwartungen gegenüber dem System:** Paul möchte ohne viel Aufwand gute Ergebnisse, will aber auch Eingriffsmöglichkeiten, um die Stundenplanung zu tunen. Erhofft sich durch Software eine Entlastung

Personas sollten so modelliert werden, dass sich das Entwicklungsteam ein konkretes Bild dieses gedachten Menschen machen kann

Dadurch macht man sich die Empathie der Team-Mitglieder besser zunutze

Abstrakte Anwendungsfälle „bekommen ein Gesicht“

Persona: Eigenschaften

- **Ziele**

- Was will die Persona (mithilfe des Systems) erreichen?
- Welche Aufgaben muss die Persona erledigen?
- Für was ist sie verantwortlich (Rolle, Verantwortungsbereich, Berufsbezeichnung, konkrete Tätigkeiten)?

- **Kenntnisse**

Ausbildung, Computerkenntnisse, Sprachkenntnisse, Wissen über verwandte Produkte und Altsysteme, Berufserfahrung, fachliches Wissen

- **Einstellungen**

Ängste, Sehnsüchte, Vorlieben, Abneigungen

- **Demografische Daten**

Geschlecht, Alter, ethnische Herkunft, ...

- **Erwartungen gegenüber dem System**

Erwartungen, Wünsche bezüglich des Systems, Einstellung zur Arbeit mit dem System

Zur Modellierung müssen Eigenschaften ermittelt werden, die das Verhalten und die Wahrnehmung in Bezug auf das System beeinflussen

[-> Übung-Personas]

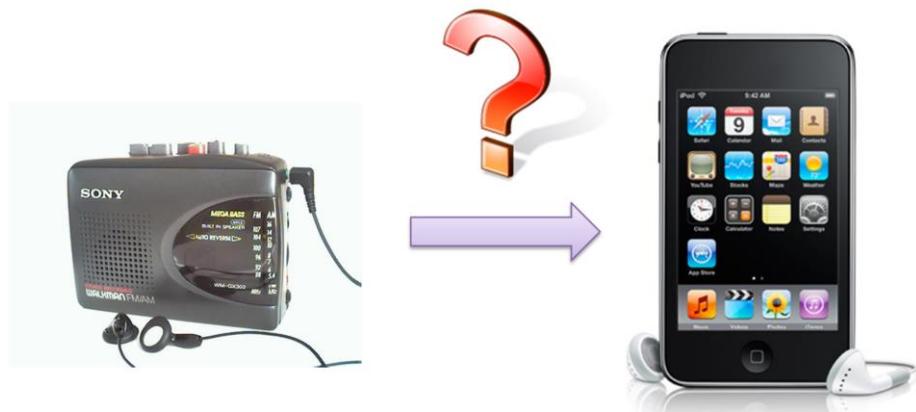
Abschnitt 1.10.3

INNOVATION ODER „MORE OF THE SAME“

09.11.2018

Software Engineering 2

128



„Wenn ich meine Kunden gefragt hätte, was sie wollen, hätten sie mir geantwortet: „Ein schnelleres Pferd““ – Henry Ford

Disruptive Innovation erreicht man nicht durch Detailsverbesserungen am status-quo.

Dem Kunden
genau das liefern,
was er sagt.

Wir wissen schon,
was gut für den
Kunden ist.

Beides falsch

Wie entsteht Innovation?

- Den Beteiligten innovative Lösungen **vorschlagen**
- Kreativität der Beteiligten **anregen**
 - Zukunftsszenarien entwerfen und durchspielen
 - Alle Beschränkungen fallen lassen
 - Metaphern suchen und explorieren
- Anforderungen lassen sich nicht einfach „erheben“

Auskennen in der aktuellen Technologie, um dem Kunden aufzeigen zu können, was alles möglich ist. Trends kennen und wissen, wohin sich die Technik entwickelt.

Abschnitt 1.10.4

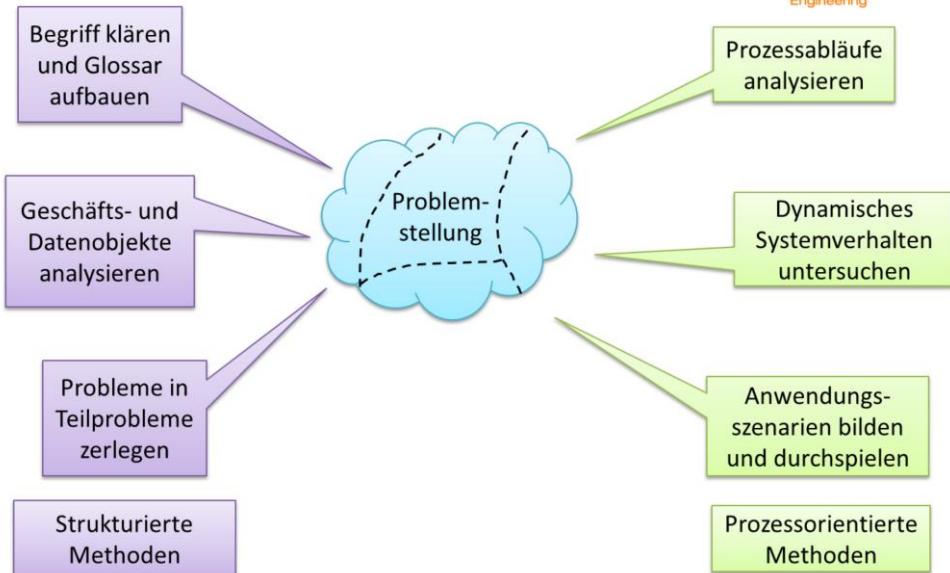
VORGEHEN UND REGELN DER ANFORDERUNGSGEWINNUNG

09.11.2018

Software Engineering 2

132

Methodisches Vorgehen



- Aussagen aus der Anforderungsermittlung enthalten oft eine Lösungsidee des Stakeholders, nicht sein zugrundeliegendes Problem.
- Der Analyst muss daraus die Essenz des Problems ableiten, um
 - das echte Problem lösen zu können,
 - zu vermeiden, alte Technologieentscheidung zu übernehmen.
- Bei der Suche nach der Essenz der geschäftlichen Aufgabe die aktuelle und die zukünftige Technologie ausblenden.

Problem vs. Lösung: Beispiele

Das Produkt muss läuten und eine blinkende Nachricht auf dem Bildschirm anzeigen, sobald eine Wetterstation keine Messdaten übermittelt.

Das Produkt muss eine Warnung ausgeben, sobald eine Wetterstation keine Messdaten übermittelt.

Der Reisende muss sein Ziel durch Berührung auf einer Karte auf dem Bildschirm auswählen.

Das Produkt muss die Eingabe des Reiseziels ermöglichen.

- Erwartungs- und Begriffs-diskrepanzen bei den Beteiligten
- Stakeholder wissen nicht, was sie wollen
- Stakeholder wissen was sie wollen, können ihre Vorstellungen aber nicht verbalisieren
- Beteiligte verfolgen verdeckte Ziele, die sie absichtlich nicht offenlegen
- Beteiligte sind auf bestimmte Lösung fixiert

Requirements Engineering ist deshalb immer auch

- Aufgabenklärung
- Risikoanalyse
- Konflikterkennung
- Konfliktauflösung
- Konsensbildung
- Anregen der Kreativität der Beteiligten
- Motivation

- Methodische Hilfsmittel für die Analyse von Rohdaten aus der Anforderungsgewinnung und dem Kickoff
- Drei bekanntere Verfahren :
 - Szenarienanalyse
 - Objekt/Nominalphrasenanalyse
 - Ereignis-Reaktions-Analyse

Grundidee: Analyse logischer Interaktionssequenzen zwischen dem Systemkontext und dem Aufgabenbereich

Geeignet für:

- Bildung und Analyse von Anwendungsszenarien
- Analyse des dynamischen Systemverhaltens
- Sekundär auch: Analyse von Prozessabläufen

Liefert:

- Szenarien und Anwendungsfälle
- Prozessablaufmodelle

Objektanalyse/Nominalphrasenanalyse

Grundidee: Analyse der **grammatikalischen Struktur** gegebener Texte

Voraussetzung: **Text vorhanden** (schriftlich oder mündlich)

Geeignet für:

- Analyse von Geschäfts- und Datenobjekten
- Aufbau von Glossaren
- Erkennen von Vorgängen

Liefert:

- Glossar
- Kandidaten für Funktionalität

Problem: Liefert große Menge schwach strukturierte Kandidaten für relevante Begriffe

Besonders nützlich für Domänenmodell/Datenmodellierung.

Durchführung

- Text grammatisch analysieren
 - Grammatisches *Subjekt*, grammatisches *Objekt* → Kandidaten für *Objekte*, *Klassen*, *Attribute* oder *Wertebereiche*
 - *Verben* beschreiben Zusammenhänge oder Handlungen:
 - Zusammenhänge → *Beziehungen*, *Attribute*
 - Handlungen → *Funktionalität*, *Verhalten*
 - *Adjektive* präzisieren Aussagen oder schränken sie ein

Durchführung II

- Fragmente **klassifizieren, ordnen, vervollständigen**
 - **Synonyme** identifizieren
 - Konkrete Objekte und Attributwerte **abstrahieren** zu Klassen und Attributen
 - Neu gewonnen Information mit bereits vorhandenen Modellteilen **verknüpfen**
- Problem der Abgrenzung von Klassen/Objekten gegen Attribute/Werte:
 - Objekt muss Identität haben
 - Attributwerte sind Daten ohne eigene Identität

Beispiel machen: Reisebüro [→Beispiel-Objektanalyse, →Beispiel-Objektanalyse-Lösung]

Abschnitt 1.11

ANFORDERUNGSDOKUMENTATION

09.11.2018

Software Engineering 2

142

Qualitätskriterien für Anforderungen



Adäquat: Beschreiben, was der Kunde will/braucht



Vollständig: Beschreiben alles, was der Kunde will/braucht



Widerspruchsfrei: Inkonsistente Anforderungen können nicht umgesetzt werden

$$P(m) \Leftrightarrow m^{22}$$

Verständlich: Anforderungen müssen für alle Beteiligten (auch Kunden) verständlich beschrieben sein



Eindeutig: Vermeidet Fehlinterpretationen



Anforderung

Prüfbar: Sonst kann nicht festgestellt werden, ob das System den Anforderungen genügt



Risikogerecht: Je geringer das akzeptierte Risiko, desto umfangreicher die Spezifikation

Abschnitt 1.11.1

DAS ANFORDERUNGSDOKUMENT

09.11.2018

Software Engineering 2

144

Heißt auch „die Spezifikation“, „Software Requirements Specification“, „Anforderungsspezifikation“, „Lastenheft“

- Keine definitiven Vorgaben aber viele Vorschläge
- Teilweise unternehmensinterne Standards
- IEEE Standard 830-1998 enthält Gliederung, die speziell für die Dokumentation von Softwareanforderungen entwickelt wurde
- Anderer Gliederungsvorschlag ist das Volere Requirements Specification Template

Institute of Electric and Electronic Engineers: IEEE Recommended Practice for Software Requirements Specifications (IEEE Std. 830-1998)

Aufbau einer Anforderungsspezifikation

Warum Standardgliederungen?

- Anforderungsdokumente enthalten eine Vielzahl von Informationen, die **für den Leser gut strukturiert** sein müssen
- Standardgliederungen bieten **vordefinierte Kategorien**, in die Informationen eingesortiert werden können
- Standardgliederungen geben die Grobstruktur vor und erläutern, welche Inhalte in den einzelnen Kapiteln stehen sollen

Basieren auf Erfahrungen, welcher Aufbau gut zu lesen ist.

Warum Standardgliederungen?

- Die Festlegung einer standardisierten Gliederung bietet eine Vielzahl von Vorteilen:
 - Leichtere Einarbeitung neuer Mitarbeiter
 - Schnelleres Erfassen ausgewählter Inhalte
 - Ermöglichen selektives Lesen/Überprüfen von Anforderungsdokumenten
 - Ermöglichen automatische Überprüfung gewisser Qualitätsanforderungen, z.B. Vollständigkeit
 - Verbessern die Wiederverwendung von Anforderungsdokumenten

Es gibt einige Vorschläge für die Strukturierung, z.B.
Software Requirements Specification im Unified Process
Lasten- und Pflichtenheft im V-Modell(XT)

Sowie die drei, die im folgenden detaillierter betrachtet werden:

IEEE 830

Volere-Schablone

Es gibt aber noch viele weitere Varianten

Projektgeber	14. Wartungs- und Unterstützungsanforderungen
1. Zweck des Projekts	15. Sicherheitsanforderungen
2. Auftraggeber, Kunde, andere Stakeholder	16. Kulturelle und politische Anforderungen
3. Benutzer des Produkts	17. Rechtliche Anforderungen
Randbedingungen des Projekts	Projektsaspekte
4. Einschränkungen	18. Offene Punkte
5. Namenskonventionen und Definitionen	19. Standardlösungen
6. Relevante Fakten und Annahmen	20. Neue Probleme
Funktionale Anforderungen	21. Aufgaben
7. Aufgabenbereich (Scope of the work)	22. Migrationstätigkeiten
8. Systemgrenzen (Scope of the product)	23. Risiken
9. Funktionale und Daten-Anforderungen	24. Kosten
Nichtfunktionale Anforderungen	25. Benutzerdokumentation
10. Look-and-feel-Anforderungen	26. Zurückgestellte Anforderungen
11. Usability-Anforderungen	27. Lösungsideen
12. Leistungsanforderungen	
13. Operationale und Umfeld-Anforderungen	

Volere Requirements Resources: www.volere.co.uk

S. Robertson, J. Robertson: Mastering the Requirements Process, Addison-Wesley, 2006

- 3. Benutzer des Systems umfasst z.B. auch Wartungs- und Betriebspersonal
- 4. Einschränkungen sind z.B. Vorgaben zur Technologie/Bibliotheken, das Betriebsumfeld des aktuellen Systems, möglicherweise einzusetzende Standardsoftware, Zeitconstraints, Budgetconstraints
- 6. Relevante Fakten umfasst auch Business Rules
- 7. Scope of the work befasst sich mit der Anwendungsdomäne, der gegenwärtigen Situation, dem Kontext der zu erledigenden Arbeit, Arbeitsaufteilung und Business Use Case
- 8. Scope of the Product beschreibt die Abgrenzung des Teils von 7, der im Produkt behandelt werden soll
- 10. Look-and-feel: Style-Guides die einzuhalten sind u.ä.
- 11. Usability: Vorgaben zur Bedienbarkeit („soll für 11-jährige Kindern einfach zu benutzen sein“), Personalisierung, Internationalisierung, Anforderungen zur Erlernbarkeit des Systems, Verständlichkeit, Höflichkeit, Zugang für Benutzer mit Beeinträchtigungen
- 12. Leistungsanforderungen: Geschwindigkeit und Verzögerung, Sicherheitskritische Anforderungen, Genauigkeit bei Berechnungen, Verlässlichkeit/Verfügbarkeit, Robustheit und Fehlertoleranz, Kapazität, Skalierbarkeit, Langlebigkeit
- 13. Operationale und Umfeld-Anforderungen: Physisches Umfeld des Systems, Schnittstellen zu angrenzenden Systemen, Anforderungen für den Produktiveinsatz, Anforderungen an Releases
- 15. Sicherheitsanforderungen: Zugriffskontrolle, Integrität, Privacy, Auditing, Immunität gegen Angriffe
- 20. Neu aufgetretene Probleme: Probleme, die das System neu verursacht, z.B. durch Auswirkungen auf bestehende Systeme oder das Betriebsumfeld

Das Template, wie auch der IEEE-Standard werden natürlich nicht auf einen Schlag vollständig ausgefüllt, sondern nach und nach

- **Zweck des Projekts:** Warum wird das Projekt durchgeführt? Welches Problem gibt es im Geschäft des Auftraggebers und wie soll das Produkt bei der Lösung helfen?
- **Auftraggeber, Kunde, andere Stakeholder:** Wer hat ein Interesse an dem Produkt? Welche Bedürfnisse haben die Stakeholder in Bezug auf das Produkt?
- **Benutzer des Produkts:** Wer interagiert später direkt mit dem Produkt? Welche Eigenschaften und Kompetenzen haben diese Leute?

3. Benutzer des Systems umfasst z.B. auch Wartungs- und Betriebspersonal

- **Einschränkungen:** Einschränkungen sind global, sie betreffen das Produkt als Ganzes. Das Produkt muss so konstruiert werden, dass es den Einschränkungen genügt. Einschränkungen sind damit auch Requirements, allerdings von außen vorgegeben.
- **Namenskonventionen und Definitionen:** Das spezifische Vokabular des Projekts. Begriff der Problemdomäne, Akronyme und Data Dictionary
- **Relevante Fakten und Annahmen:** Relevante Fakten sind externe Faktoren mit Auswirkungen auf das Projekt, die aber in keinen anderen Abschnitt passen. Die Annahmen sind solche, die das Projektteam macht. Sie dienen als Warnung an das Management über Faktoren, die den Projekterfolg beeinflussen können.

4. Einschränkungen sind z.B. Vorgaben zur Technologie/Bibliotheken, das Betriebsumfeld des aktuellen Systems, möglicherweise einzusetzende Standardsoftware, Zeitconstraints, Budgetconstraints

6. Relevante Fakten umfasst auch Business Rules. Annahmen beschäftigen sich oft mit dem Umgebung, in der das Produkt arbeiten wird.

- **Aufgabenbereich:** Wie ist der Aufgabenbereich abgegrenzt? Mit welchen Geschäftsbereich befassen wir uns, wie passt der Aufgabenbereich in die Umgebung?
- **Systemgrenzen:** Was ist der Umfang des Produkts? Oftmals ein Kontextmodell auf Systemebene oder ein Use-Case-Diagramm plus Liste der Product-Use-Cases
- **Funktionale und Daten-Anforderungen:** Die atomaren Anforderungen, die beschreiben, was das Produkt tun soll
- **Nicht-funktionale Anforderungen (10-17):** Siehe Abschnitt atomare Anforderungen

7. Scope of the work befasst sich mit der Anwendungsdomäne, der gegenwärtigen Situation, dem Kontext der zu erledigenden Arbeit, Arbeitsaufteilung und Business Use Cases

8. Scope of the Product beschreibt die Abgrenzung des Teils von 7, der im Produkt behandelt werden soll

10. Look-and-feel: Style-Guides die einzuhalten sind u.ä.

11. Usability: Vorgaben zur Bedienbarkeit („soll für 11-jährige Kindern einfach zu benutzen sein“), Personalisierung, Internationalisierung, Anforderungen zur Erlernbarkeit des Systems, Verständlichkeit, Höflichkeit, Zugang für Benutzer mit Beeinträchtigungen

12. Leistungsanforderungen: Geschwindigkeit und Verzögerung, Sicherheitskritische Anforderungen, Genauigkeit bei Berechnungen, Verlässlichkeit/Verfügbarkeit, Robustheit und Fehlertoleranz, Kapazität, Skalierbarkeit, Langlebigkeit

13. Operationale und Umfeld-Anforderungen: Physisches Umfeld des Systems, Schnittstellen zu angrenzenden Systemen, Anforderungen für den Produktiveinsatz, Anforderungen an Releases

15. Sicherheitsanforderungen: Zugriffskontrolle, Integrität, Privacy, Auditing, Immunität gegen Angriffe

- **Offene Punkte:** Gibt es Probleme, die bisher ungelöst sind?
- **Standardlösungen:** Gibt es verfügbare fertige Lösungen? Welche Eigenschaften haben sie, inwiefern erfüllen sie die Anforderungen?
- **Neue Probleme:** Änderungen an der bestehenden Ordnung der Dinge können negative Effekte haben. Welche Probleme werden durch die Installation des neuen Produkts geschaffen?
- **Aufgaben:** Welche Schritte sind nötig, um das Produkt auszuliefern?

20. Neu aufgetretene Probleme: Probleme, die das System neu verursacht, z.B. durch Auswirkungen auf bestehende Systeme oder das Betriebsumfeld. Beispiel: Erhöhte Last auf der Datenbank, durch neue Queries für neue Reporting-Funktionen

- **Migrationstätigkeiten:** Welche Tätigkeiten sind erforderlich, um bei der Installation die Einsatzbereitschaft des Produkts herzustellen? Sind z.B. Datenbanken zu migrieren?
- **Risiken:** Alles, was den Projekterfolg gefährdet.
- **Kosten:** Was kostet es, das Produkt herzustellen?
- **Benutzerdokumentation:** Was muss an Unterlagen, Handbüchern, Training, ... für die Benutzer erstellt bzw. vorbereitet werden?

- **Zurückgestellte Anforderungen:** Anforderungen, die nicht in die aktuell zu erstellende Produktversion integriert werden können.
- **Lösungsideen:** Lösungsideen oder Vorschläge, die im RE-Prozess auftauchen oder vorgeschlagen werden, werden in diesem Bereich gesammelt.