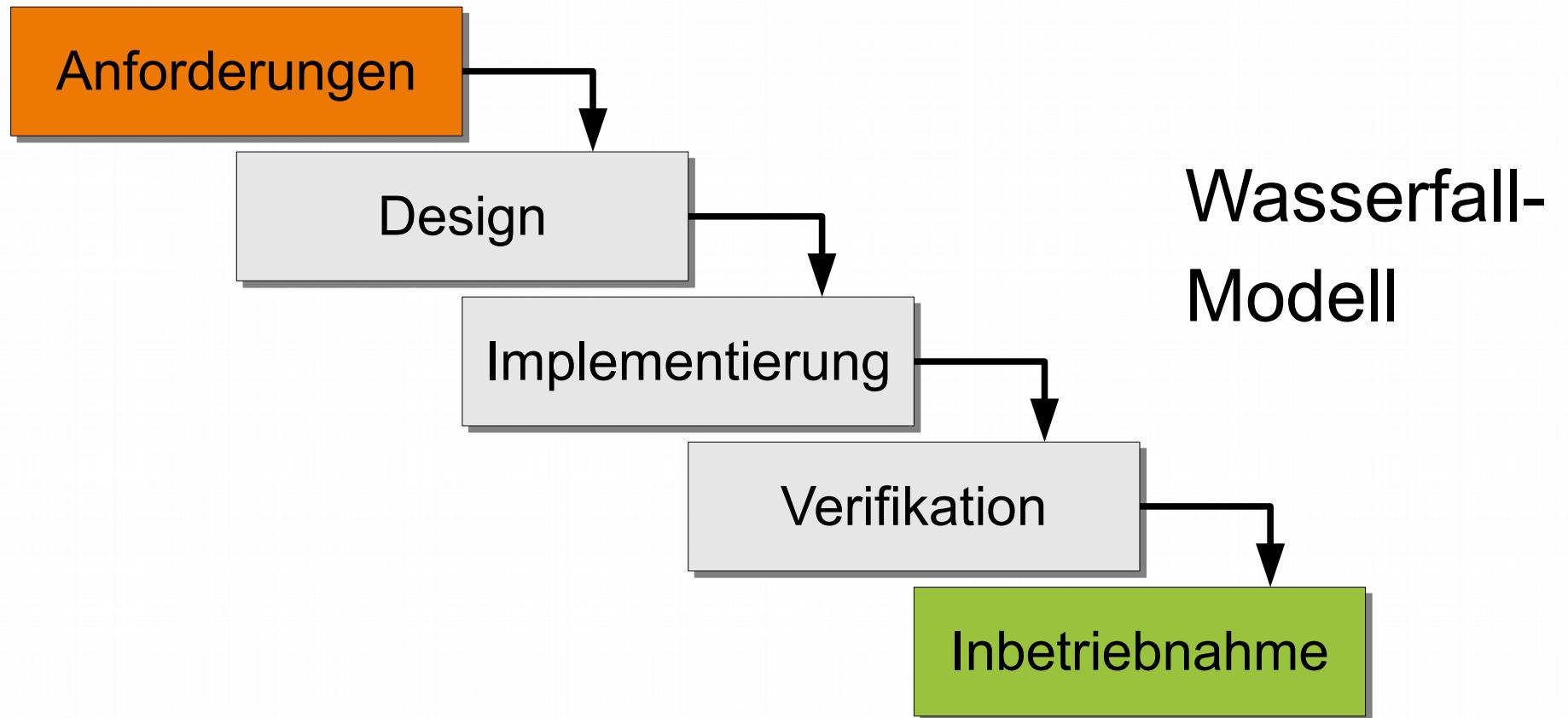


Anforderungs- analyse

Von der Idee zum Code

Klassische Softwareentwicklung

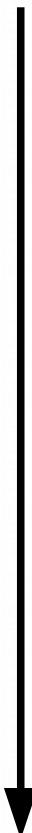


Aus Kundensicht

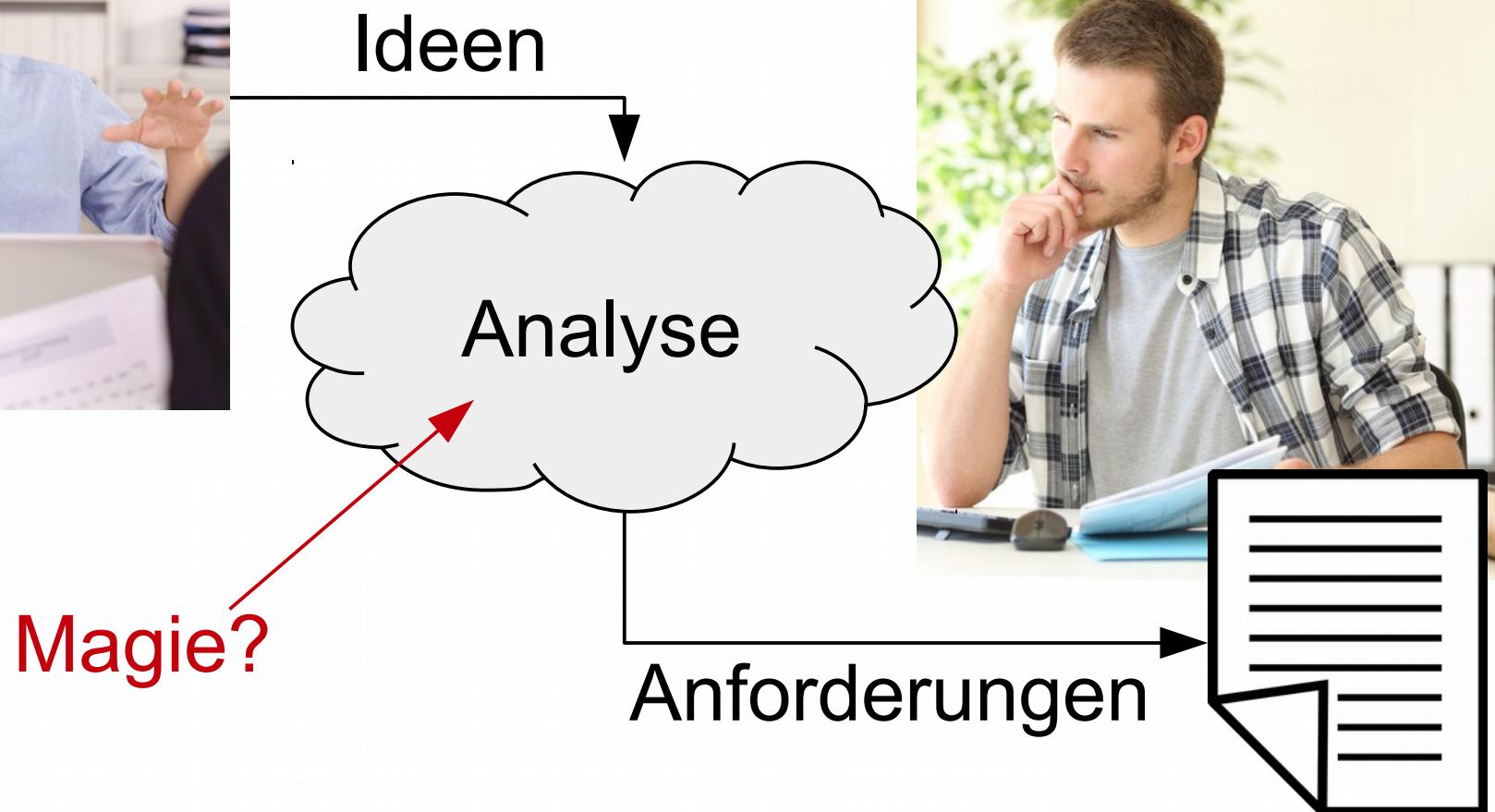
Anforderungen

Obskure
Vorgänge

Inbetriebnahme



Anforderungsanalyse



Anforderungsanalyse

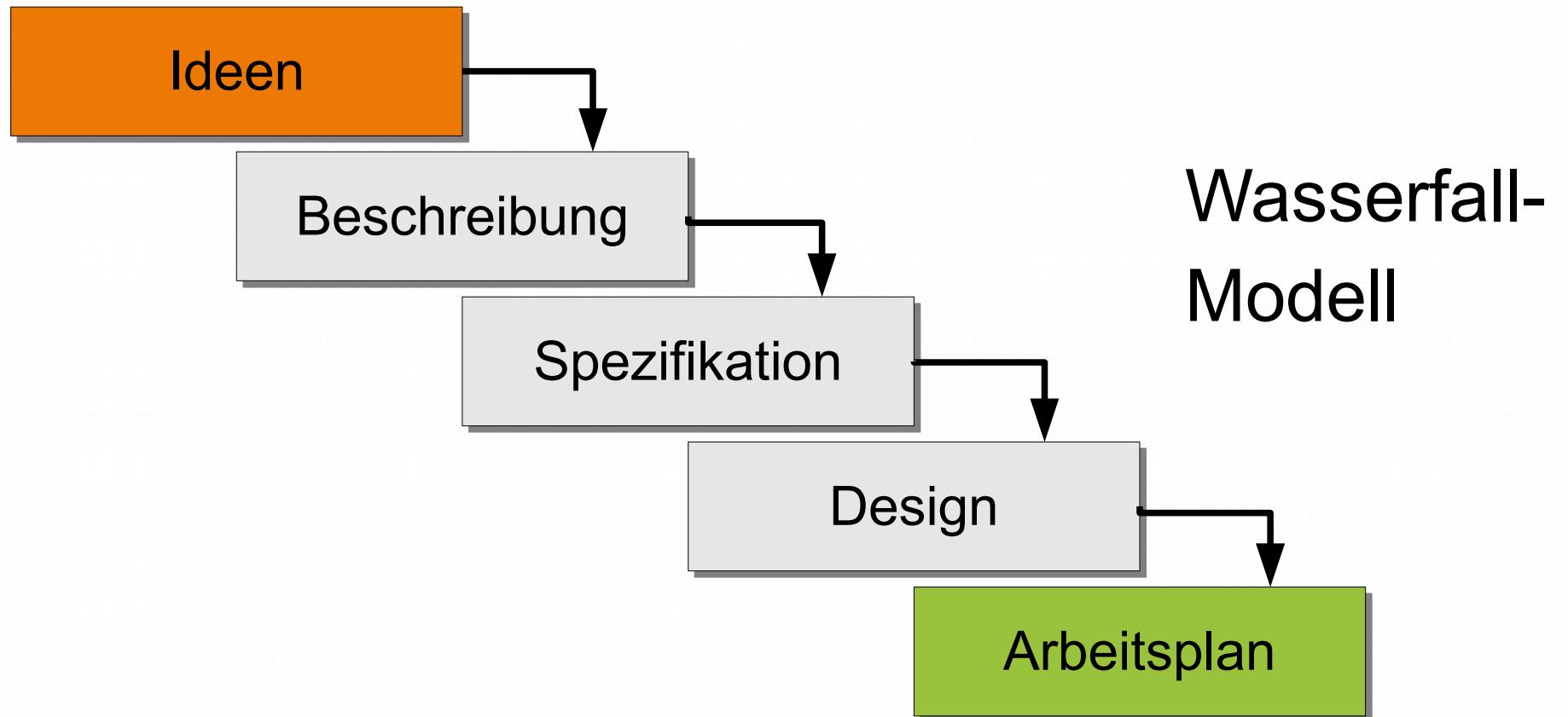
Erstellt aus den Ideen, Wünschen und Bedürfnissen des Kunden das Anforderungsdokument



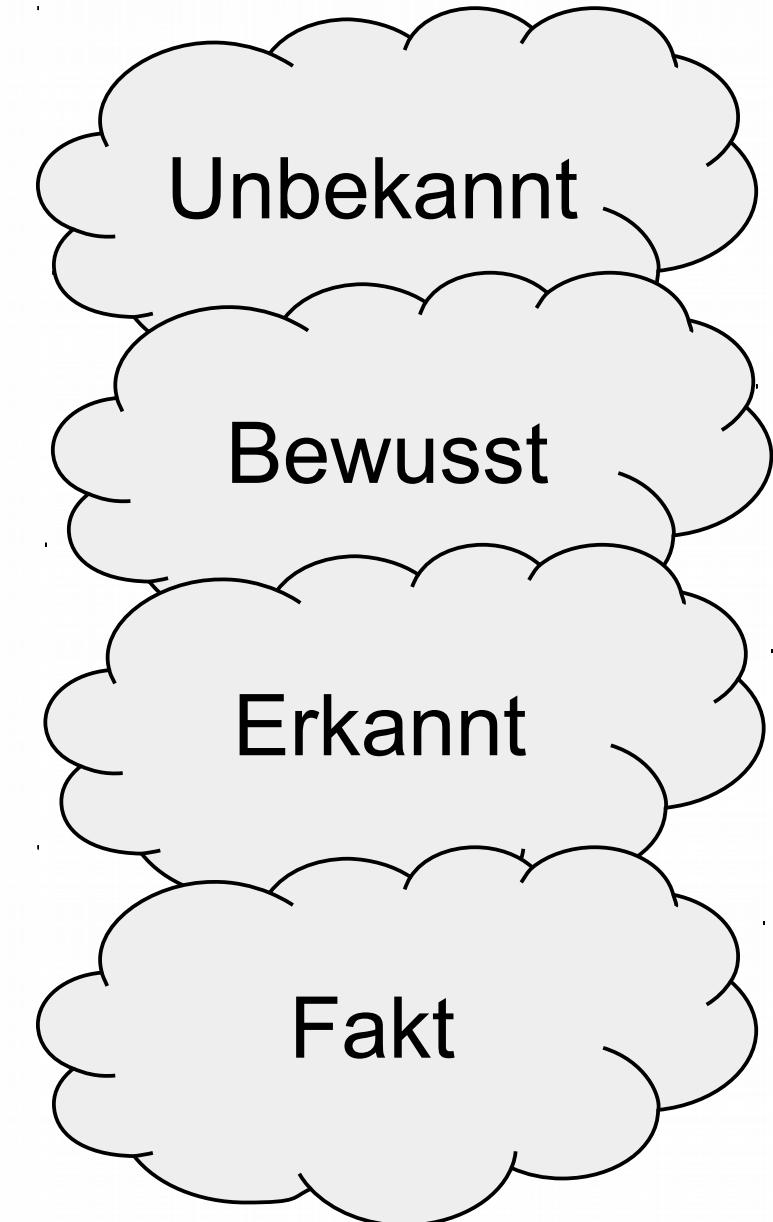
Am besten mit einem Vorgehen, das strukturiert und wiederholbar ist

Idealerweise kann dieses Vorgehen jeder direkt oder mit kurzer Schulung anwenden

Klassische Anforderungsanalyse



Anforderungsanalyse generell

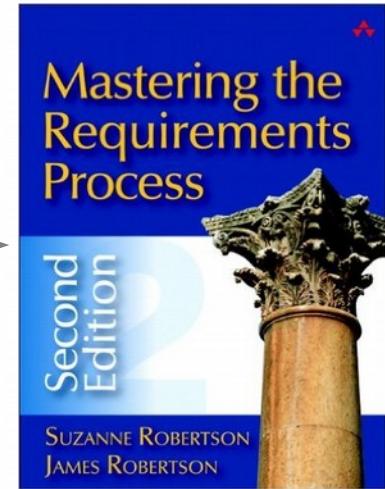


- Jeder Arbeitsschritt sollte
 - Das Wissen eine Erkenntnisstufe weiterbringen
- Jedes Ergebnis (Artefakt) sollte
 - Einzigartig in seiner Erkenntnisstufe sein
 - Die Artefakte der vorigen Stufe nicht duplizieren
- Formulierte Erkenntnisse sollten immer konkreter werden

Mögliche Vorgehen

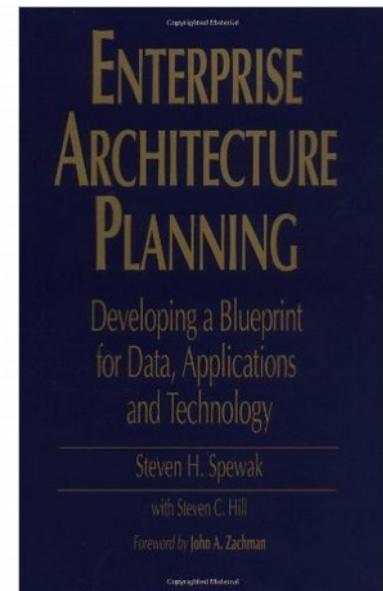
- Volere
- Zachman-Framework
- Freies Kundeninterview
- Beobachtung
- Workshops
- Umfragen
- System-, Oberflächen- oder Dokumentanalyse

- Kostenpflichtige Vorlagen („Template“)
 - Von der Atlantic Systems Guild
 - Tom DeMarco, Tim Lister, etc.
 - Umfang von 60-90 Seiten
 - Mit begleitendem Buch
 - Mit zubuchbarem Training
-
- Fazit: Alles drin, aber sehr komplex



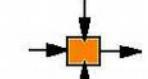
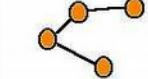
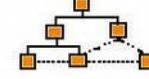
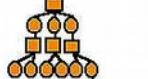
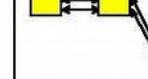
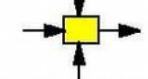
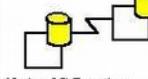
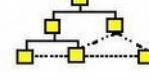
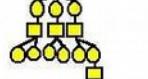
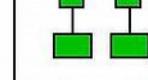
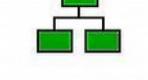
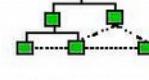
Zachman-Framework

- Domänenneutraler Ordnungsrahmen zur Entwicklung von Informationssystemen
- Generisches Konzept mit geringem Formalisierungsgrad
- Kein definierter Prozess
- Konzentriert sich auf „Rollen“ und „Perspektiven“
 - Was, Wie, Wo, Wer, Wann und Warum



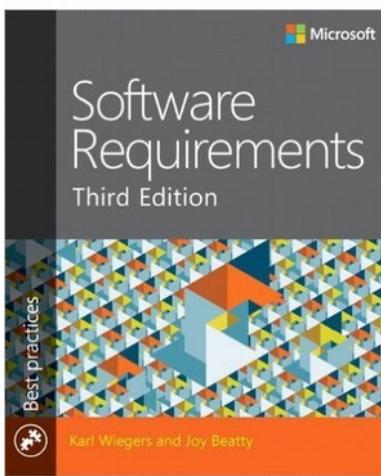
Zachman-Framework

ENTERPRISE ARCHITECTURE - A FRAMEWORK™

	DATA <i>What</i>	FUNCTION <i>How</i>	NETWORK <i>Where</i>	PEOPLE <i>Who</i>	TIME <i>When</i>	MOTIVATION <i>Why</i>	
SCOPE (CONTEXTUAL)	List of Things Important to the Business 	List of Processes the Business Performs 	List of Locations in which the Business Operates 	List of Organizations Important to the Business 	List of Events Significant to the Business 	List of Business Goals/Strat 	SCOPE (CONTEXTUAL) Planner
	Ent/ITV = Class of Business Thing	Function = Class of Business Process	Node = Major Business Location	People = Major Organizations	Time = Major Business Event	Ends/Mans=Major Bus. Goal/Critical Success Factor	
ENTERPRISE MODEL (CONCEPTUAL)	e.g. Semantic Model 	e.g. Business Process Model 	e.g. Logistics Network 	e.g. Work Flow Model 	e.g. Master Schedule 	e.g. Business Plan 	ENTERPRISE MODEL (CONCEPTUAL) Owner
	Ent = Business Entity Reln = Business Relationship	Proc. = Business Process IO = Business Resources	Node = Business Location Link = Business Linkage	People = Organization Unit Work = Work Product	Time = Business Event Cycle = Business Cycle	End = Business Objective Means = Business Strategy	
SYSTEM MODEL (LOGICAL)	e.g. Logical Data Model 	e.g. "Application Architecture" 	e.g. "Distributed System Architecture" 	e.g. Human Interface Architecture 	e.g. Processing Structure 	e.g., Business Rule Model 	SYSTEM MODEL (LOGICAL) Designer
	Ent = Data Entity Reln = Data Relationship	Proc = Application Function IO = User Views	Node = IS Function (Processor, Storage etc.) Link = Line Characteristics	People = Role Work = Deliverable	Time = System Event Cycle = Repeating Cycle	End = Structural Assertion Means = Action Assertion	
TECHNOLOGY MODEL (PHYSICAL)	e.g. Physical Data Model 	e.g. "System Design" 	e.g. "System Architecture" 	e.g. Presentation Architecture 	e.g. Control Structure 	e.g. Rule Design 	TECHNOLOGY CONSTRAINED MODEL (PHYSICAL) Builder
	Ent = Segment/Table/etc. Reln = Pointer/Key/etc.	Proc = Computer Function IO = Screen/Device Formats	Node = Hardware/System Software Link = Line Specifications	People = User Work = Screen Format	Time = Execute Cycle = Component Cycle	End = Condition Means = Action	
DETAILED REPRESENTATIONS (OUT-OF-CONTEXT)	e.g. Data Definition 	e.g. Program 	e.g. "Network Architecture" 	e.g. Security Architecture 	e.g. Timing Definition 	e.g. Rule Specification 	DETAILED REPRESENTATIONS (OUT-OF-CONTEXT) Sub-Contractor
	Ent = Field Reln = Address	Proc = Language Stmt IO = Control Block	Node = Addresses Link = Protocols	People = Identity Work = Job	Time = Interrupt Cycle = Realtime Cycle	End = Sub-condition Means = Step	
FUNCTIONING ENTERPRISE	e.g. DATA	e.g. FUNCTION	e.g. NETWORK	e.g. ORGANIZATION	e.g. SCHEDULE	e.g. STRATEGY	FUNCTIONING ENTERPRISE

Zachman Institute for Framework Advancement - (810) 231-0531

Freies Kundeninterview



- Mit dem Kunden über sein Projekt sprechen
- Möglichst Agenda formulieren
 - Thema im Auge behalten
- Aktives Zuhören
 - Bei Unklarheiten nachfragen
 - Aussagen mit eigenen Worten wiederholen (umschreiben)
- Eigene Ideen und Gedanken anbieten (Zwei-Wege-Gespräch)

Ziel des Interviews

- Durch gezielte Fragen den Kunden (Stakeholder) dazu zu bringen, alle Anforderungen anzusprechen
- Alle Anforderungen in einem Dokument zusammenzutragen
- Das Anforderungsdokument



Tipps für die Durchführung

- Fragen, Grafiken und Modelle vorbereiten
 - Auch wenn sie falsch sind
 - Richtigstellen ist einfacher als selbst Erstellen
 - Auf Bestehendem aufbauen setzt bereits viele Rahmenbedingungen
- Oft nach dem „Warum“ fragen
 - Five-Why-Methode
 - Nicht übertreiben (Nervensäge)

<https://de.wikipedia.org/wiki/5-Why-Methode>

Tipps für die Durchführung

- Die gleiche Frage anders formuliert nochmal stellen (evtl. mit zeitlichem Versatz)
 - Triangulation der Wahrheit
- Aussagen des Kunden leicht verfälscht wiederholen bzw. umschreiben
 - Richtigstellung ermöglicht ebenfalls Triangulation
 - Nicht zu oft anwenden, sonst geht Vertrauen verloren



Tipps für die Durchführung

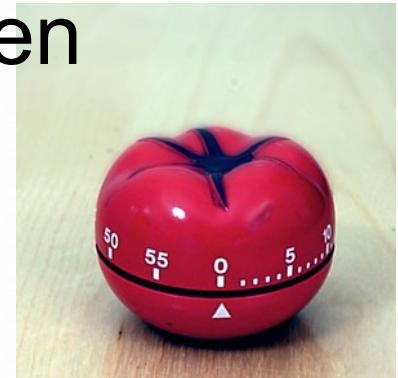
- Konkrete Beispiele geben lassen, v. a. für
 - Formeln
 - Regeln
 - Zahlen- oder textbasierte Daten
 - Die Beispiele live durchspielen und notieren
 - Gute Grundlage für automatisierte Tests
- Kunde: „Das Kleiner-als-Zeichen soll Werte herausfiltern, die größer oder gleich groß sind“
 - Ihr: „<42 lässt also nur Werte bis 41 durch?“
 - Kunde: „ja, genau!“

Tipps für die Durchführung

- Ein dedizierter Anforderungsanalyst
 - Ein Gehirn, eine Verantwortung
- Ein fachkundiger Stenografierer darf und sollte unterstützen
 - Sollte nicht inhaltlich eingreifen
- Rohe Notizen unmittelbar an den Kunden herausgeben
 - Ersetzen die Reinfassung der Ergebnisse nicht
 - Semantische Anmerkungen mit Symbolen codieren

Tipps für die Durchführung

- Zeitlich beschränkte Gespräche (max. 2 h)
 - Nach 3 h ist die erforderliche Konzentration weg
 - Lieber mehrere Termine als ein langer
- Zur Not (langer Anreiseweg) viele Pausen
 - Pomodoro-Technik: Halbstunden-Zyklen
 - 25 Minuten konzentriertes Arbeiten
 - 5 Minuten Pause
 - Nach 3-5 Zyklen eine längere Pause
- In den Pausen an die frische Luft



<https://de.wikipedia.org/wiki/Pomodoro-Technik>

Hohe Zahl an Ideen pro Minute?



Ideen
Ideen
Ideen



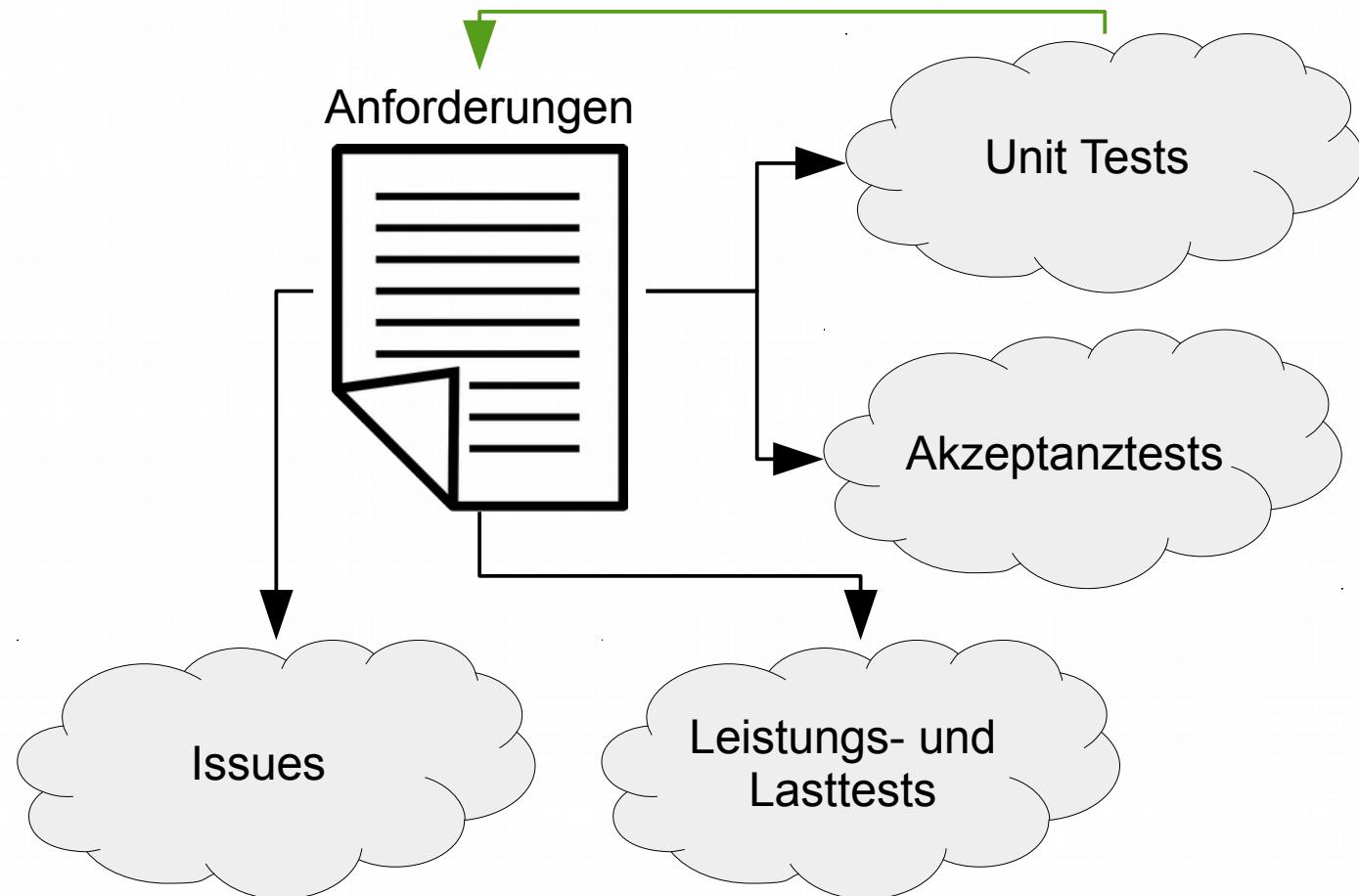
- Audio-Aufnahme
- Vorher fragen!
- Unbedingt nacharbeiten!
 - transkribieren
- Keine Ausrede für Nicht-Zuhören
- Nicht im Gespräch zurückspulen

Nach dem Interview

- Mitschrieb „ins Reine schreiben“
- Zeit mit den Anforderungen verbringen
- Ein konsistentes Bild sollte sich formen
 - Wenn nicht: Noch ein Interview ansetzen
- Die Anforderungsanalyse erst beenden, wenn die Anforderungen „klar“ sind
 - Sie werden nicht vollständig korrekt sein, das bekommt man aber erst „hinterher“ heraus
- Das Anforderungsdokument enthält die Essenz aus den Interviews

Anforderungsdokument

- Muss Folgen haben
- Direkt ableitbare Artefakte



Kriterien für Anforderungen

- AN-10. Vollständig
- AN-20. Eindeutig definiert / abgegrenzt
- AN-30. Verständlich beschrieben
- AN-40. Atomar
- AN-50. Identifizierbar
- AN-60. Einheitlich dokumentiert
- AN-70. Nachprüfbar
- AN-80. Vorwärts- und Rückverfolgbar
- AN-90. Konsistent

Kriterien für Anforderungen

- AN-10. Vollständig
 - Alle Anforderungen explizit beschrieben
 - Keine impliziten Annahmen
 - Offen für spätere Ergänzungen
- AN-20. Eindeutig definiert / abgegrenzt
 - Präzise Definitionen
 - Mißverständliche und unscharfe Begriffe vermeiden
 - Gute Heuristik: Kann ich einen automatisierten Test dafür schreiben?

Kriterien für Anforderungen

- AN-30. Verständlich beschrieben
 - Anforderungen lesbar und verstehtbar
 - Mit vertretbarem Aufwand
 - Von Entwicklern und Kunden gleichermaßen
- AN-40. Atomar
 - Nur eine Anforderung pro Satz oder Abschnitt
 - Atomizität = Entscheidbarkeit der Anforderung
 - Ja/Nein-Antwort auf die Frage „Anforderung erfüllt?“
 - Erfordert Trennen verschiedener Themen
 - Ähnlich der Separation of Concerns im Sourcecode

Kriterien für Anforderungen

- **AN-50. Identifizierbar**

- Jede Anforderung eindeutig identifizierbar
- Beispielsweise Anforderungs-ID
- Eventuell durch die Issue-Nummer vermittelbar
 - Wenn Anforderung ↔ Issue 1:1-Beziehung

- **AN-60. Einheitlich dokumentiert**

- Anforderungen alle im gleichen Dokument
- Durchgängige Struktur

Kriterien für Anforderungen

• AN-70. Nachprüfbar

- Verknüpft mit prüfbaren Abnahmekriterien
- Mit Quellenangabe (woher kam die Anforderung?)
- Abnahme = der Kunde akzeptiert die Software
- Abnahmekriterien als Akzeptanztest automatisieren
- Während der Entwicklung ständig prüfen (lassen)
 - Definition of Done: u.a. passender Akzeptanztest grün

Kriterien für Anforderungen

- AN-80. Nachvollziehbarkeit (Traceability)
- AN-81. Vorwärtsverfolgbar
 - Nachverfolgbar, ob Anforderung vollständig erfüllt
 - Anforderung → Funktionalität (Code)
- AN-82. Rückverfolgbar
 - Funktionalität (Code) → Anforderung
 - Erkennbar, aufgrund welcher Anforderung erstellt
 - Anforderungs-ID im Code und in den Tests
 - Sehr wichtig für die Wartungsphase
 - Nach Abnahme

Kriterien für Anforderungen

- AN-90. Konsistent
 - Untereinander widerspruchsfrei
 - Insbesondere bei mehreren Stakeholdern
 - Stakeholder → Quelle von Anforderungen
 - Explizite und implizite Widersprüche
 - Beispiel für impliziten Widerspruch:
 - Soll auf Raspberry Pi laufen und 10 Mio. Anfragen pro Sekunde verarbeiten

Kriterien für Anforderungen

- AN-10. Vollständig
- AN-20. Eindeutig definiert / abgegrenzt
- AN-30. Verständlich beschrieben
- AN-40. Atomar
- AN-50. Identifizierbar
- AN-60. Einheitlich dokumentiert
- AN-70. Nachprüfbar
- AN-80. Vorwärts- und Rückverfolgbar
- AN-90. Konsistent

Entstehendes Artefakt

- Liste mit Anforderungen
 - (+) Einzeln identifizierbar
 - (-) Ungeordnet
 - (-) Unzusammenhängend
- Zielsetzung:
 - Sammlung aller Anforderungen
 - Erster Überblick über die Domäne
 - Aussagen über die erwartete Projektgröße

Strukturierung der Anforderungen

- Beziehungen zwischen Anforderungen:
 - S-10. Abhängigkeiten
 - S-20. Zusammengehörigkeit
 - S-30. Rollenbezogenheit
- Typ der Anforderung:
 - S-40. Funktionale vs. Nicht-funktionale Anforderungen
- Motivation der Anforderung:
 - S-50. Fachlich vs. Technisch motivierte Anforderungen

S-10. Abhängigkeiten

- Nur wenige Anforderungen sind vollständig unabhängig
- Zu prüfen ist:
 - Gibt es Voraussetzungen für die Anforderung?
 - Erstellt eine topologische Sortierung
 - Bedingen sich Anforderungen gegenseitig?
 - Zirkuläre Abhängigkeit
 - Nicht einzeln umsetzbar

S-20. Zusammengehörigkeit

- Manche Anforderungen liegen semantisch nahe beieinander
 - „fachlich-logischer“ Zusammenhang
- Sollten im Verbund umgesetzt werden
- Bleiben aber eigenständig und können auch eigenständig umgesetzt werden

S-30. Rollenbezogenheit

- Jede Benutzerrolle hat ihre eigene Sicht auf die Anforderungen
- Typische Benutzerrollen
 - Administrator, Benutzer, Guest
- Anforderungen nach Rollenzugehörigkeit gruppieren
- Ziel: Benutzerspezifische Teilsysteme erstellen können
 - Beispiel Internetauftritt: Guest muss vorhanden sein, Administrator kann (eventuell) warten

S-40. Funktional und Nicht-funktional

- Funktionale Anforderungen
 - Definieren das Verhalten des Systems
 - Was soll das System machen?
 - Eingabe → Verhalten → Ausgabe
- Nicht-funktionale Anforderungen
 - Definieren die Eigenschaften des Systems
 - Wie soll das System sein?
 - Eher Qualitätsmerkmale
 - Leistungscharakteristik (z.B. Skalierbarkeit / Benutzer)
 - Ressourcenbedarf während des Betriebs (z.B. Speicherverbrauch)



S-50. Fachlich vs. Technisch

- Fachliche Anforderung = Aus der „Domäne“ stammend
- Technische Anforderung = Durch den aktuellen Stand der Technik bedingt
 - Vergleiche den Begriff der „Pure Fabrication“
- Domäne und Technik ändern sich mit jeweils eigenem Tempo
- Ziel: Anforderungen aus Domäne unabhängig von technischer Umsetzung realisieren
 - Erklärtes Ziel der „Clean Architecture“

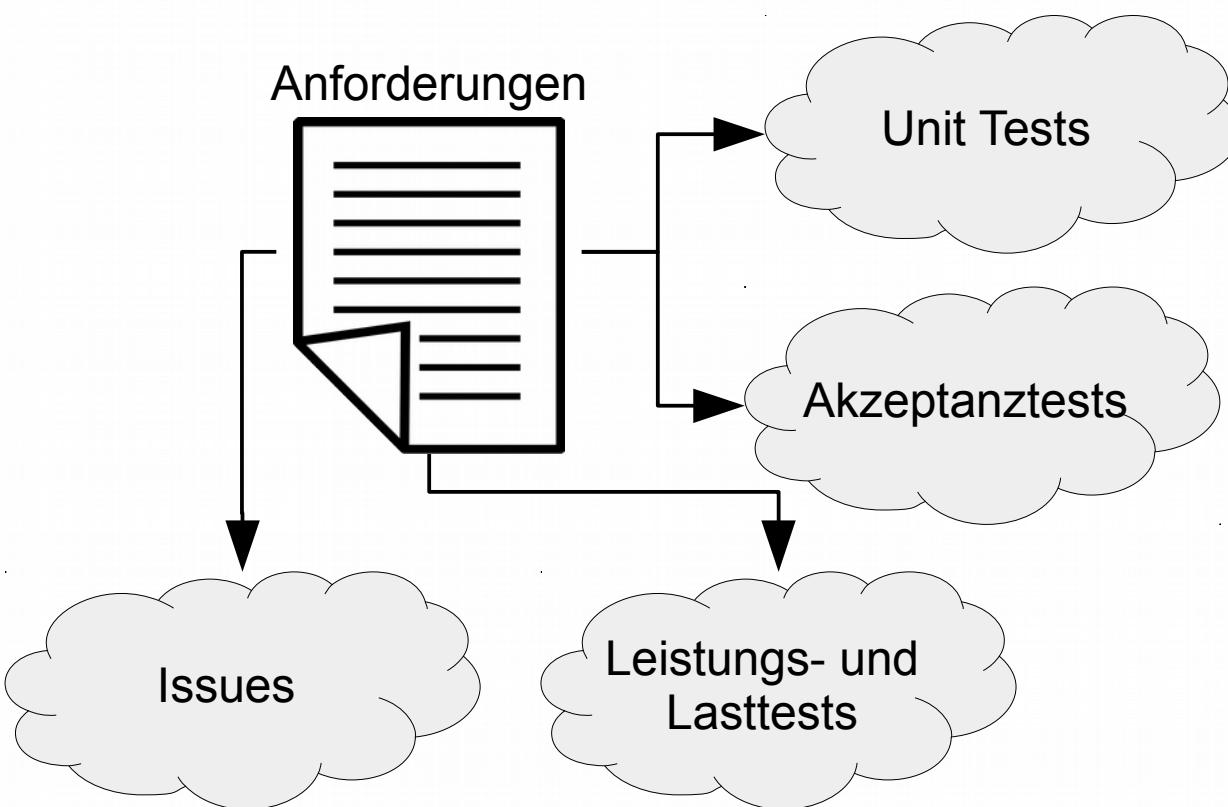
S-50. Fachlich vs. Technisch

- Brauchbare Heuristik zur Unterscheidung:
 - „Gilt diese Anforderung immer noch, wenn die Arbeit nicht von Computern, sondern von Menschen mit Stift und Papier ausgeführt würde?“
 - Ja → Domäne
 - Nein → Technische Anforderung
- Beispiele:
 - „Jedes Buch besitzt eine eindeutige Identifikationsnummer“
 - „Der Barcode befindet sich im unteren Drittel des Buchrückens“

Entstehendes Artefakt

- Topologisch sortierter Abhängigkeitsgraph
- Gruppen- bzw. Regionsbildung (Sub-Graphen)
- Zielsetzung:
 - Unmögliche Konstellationen ausschließen
 - „Nebenwirkungen“ aufzeigen
 - Identifikation von Subsystemen bzw. Modulen
 - Bei serviceorientierter Architektur (SOA) v.a. Services
 - Erwartete Nachweismittel festlegen

Nachweismittel von Anforderungen



- Funktionale A.
 - Unit Test
- Nicht-funktionale A.
 - Leistungstest
 - Lasttest
- Fachliche A.
 - Akzeptanztest
- Technische A.
 - Integrationstest

Bewertung der Anforderungen

- Jede Anforderung sollte fünf Qualitätsmerkmale aufweisen:
 - Q-10. Korrekt
 - Q-20. Machbar
 - Q-30. Notwendig
 - Q-40. Priorisiert
 - Q-50. Nutzbringend

Bewertung der Anforderungen

- Q-10. Korrekt
 - Untereinander widerspruchsfrei
 - Vom Kunden bestätigt
- Q-20. Machbar
 - unter Berücksichtigung der anderen Anforderungen umsetzbar
 - Mit dem aktuellen Stand der Technik realisierbar

Bewertung der Anforderungen

- Q-30. Notwendig
 - Vom Kunden explizit gefordert
 - Kein „Nice-to-have“-Feature
- Q-40. Priorisiert
 - Einschätzung der Wichtigkeit der Anforderung
 - Erwarteter Business Value
 - Eventuell auch nur Häufigkeit der Verwendung
 - Vom Kunden vorgenommen
 - Ziel: „Business Value First“-Ansatz

Bewertung der Anforderungen

- Q-50. Nutzbringend
 - Produktives System auch bei nur teilweiser Umsetzung
 - Durch Abhängigkeitsanalyse müssen eventuell weniger hochpriore Anforderungen trotzdem früh umgesetzt werden
 - Ziel: Inkrementell-iterative Systemerstellung
 - Einsatzbereites System am Ende jeder Iteration
 - Pro Iteration neue Funktionalität mit zusätzlichem Nutzen

Entstehendes Artefakt

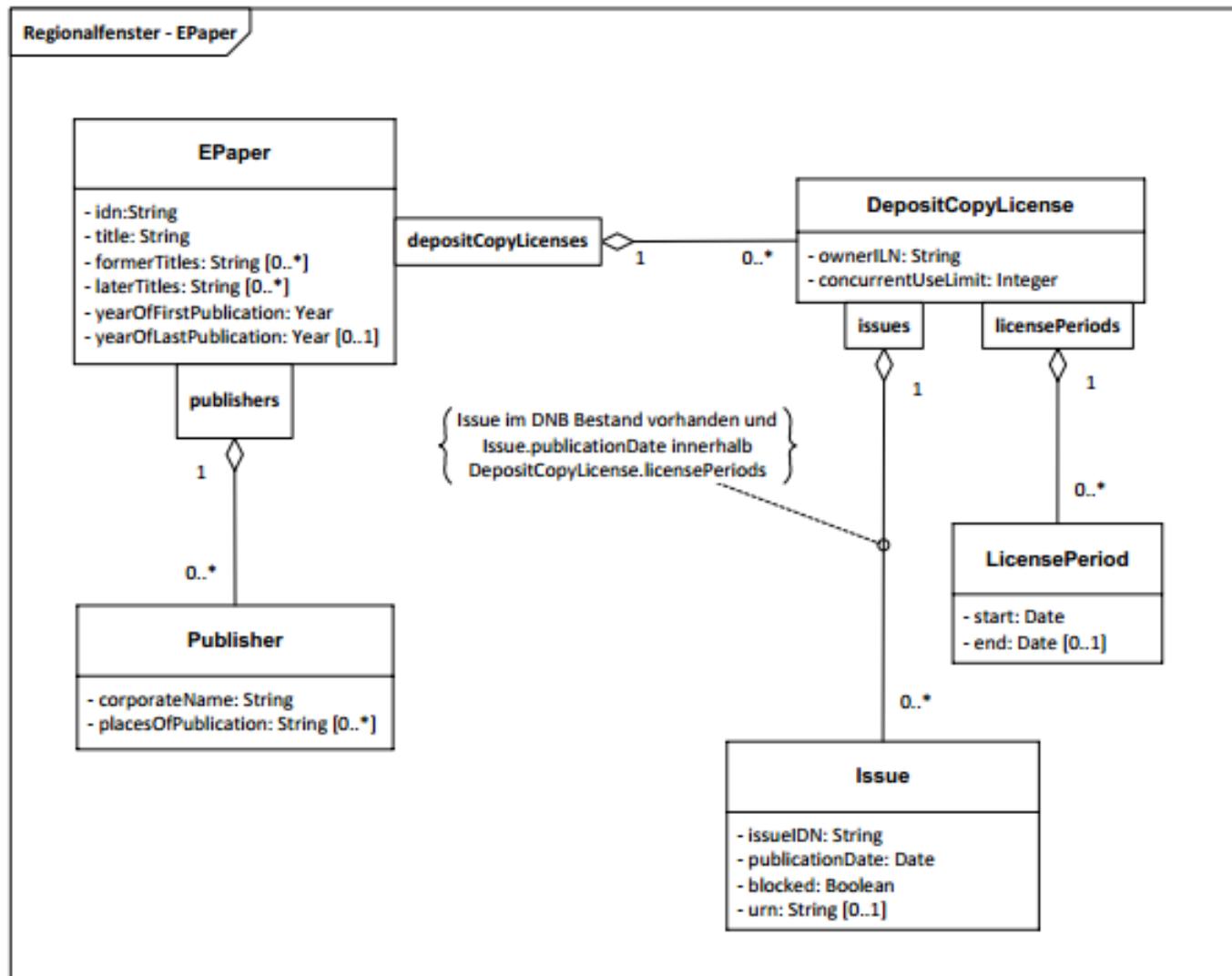
- Topologisch sortierter Abhängigkeitsgraph
- Priorisierung der Einstiegsknoten
- Zielsetzung:
 - Maximaler Business Value in minimaler Zeit
 - Dauerhafte Einsatzbereitschaft
 - Teilweise Umsetzung möglich

Grundlegende Regeln

- Jedes Anforderungsartefakt muss konkreter sein als die Vorgänger
 - Analysearbeit bewegt das Projekt Richtung Code
- Anforderungen müssen referenzierbar sein
 - Nachvollziehbarkeit als Gegenmittel gegen Chaos
- Jede Anforderung muss konkrete und nachvollziehbare Artefakte im Projekt produzieren
 - Automatisierte Tests
 - Manuelle Überprüfungen
 - Source Code

Implementierung mit Freiheitsgrad

1 Fachliches Datenmodell



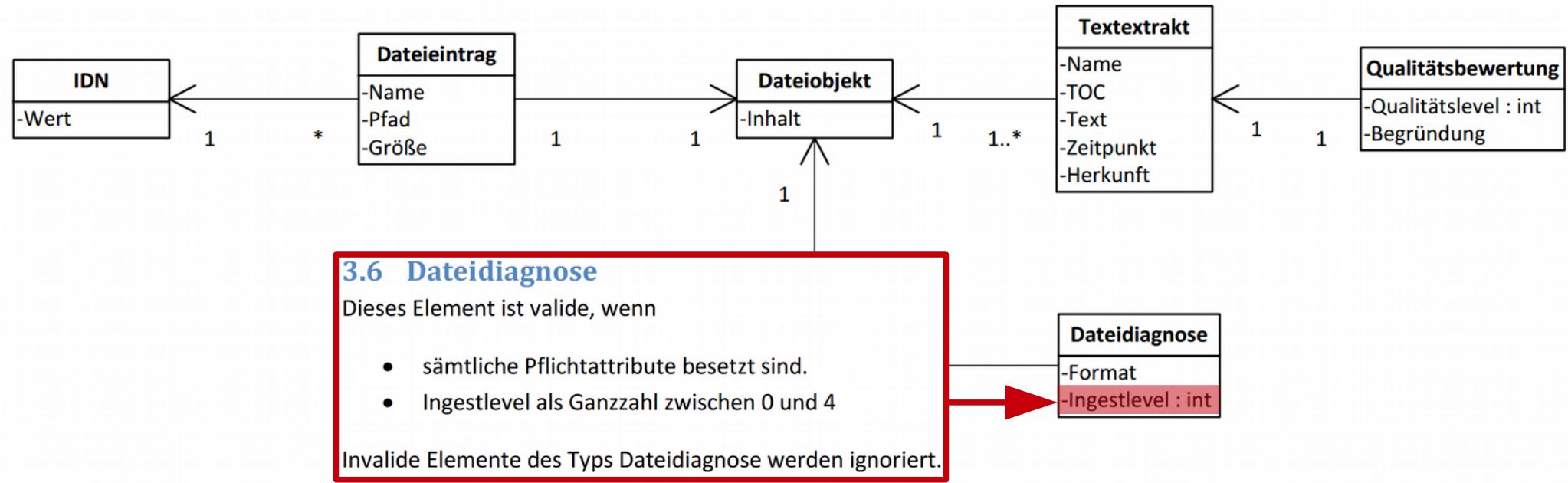
- Möglichst keine konkreten Datentypen während der Anforderungsanalyse
- Hier evtl. zu konkrete Multiplizitäten

Implementierung mit Freiheitsgrad

- Anforderungsanalyse auf Ebene der Implementierung nimmt viel Spielraum weg
- Zwei Arten von Spielraum
 - Schlechter Spielraum: Exakte Umsetzung fachlich notwendig. Abweichungen wären schädlich.
 - Whitebox-Analyse
 - Guter Spielraum: Umsetzungsergebnis benötigt, Umsetzungsweg nicht.
 - Blackbox-Analyse
- Darauf achten, dass der gute Spielraum bestehen bleibt

Freiheitsgrade zurückgewinnen

1 Fachliches Datenmodell



- Ingestlevel ist kein „int“, sondern eine „Ganzzahl zwischen 0 und 4“
- String → Zeichenkette, int → Ganzzahl oder Zahl

Fachmodell mit Freiheitsgrad

1.10 Sortierung

Beschreibt Sortierangaben, die mit der Suchanfrage übermittelt werden.

Eigenschaft	Typ	Beschreibung
Sortierschlüssel	Zeichenkette	Enthält ein oder mehrere Suchbegriffe
Sortierreihenfolge	Zeichenkette	Kennzeichnung aufsteigender oder absteigender Reihenfolge

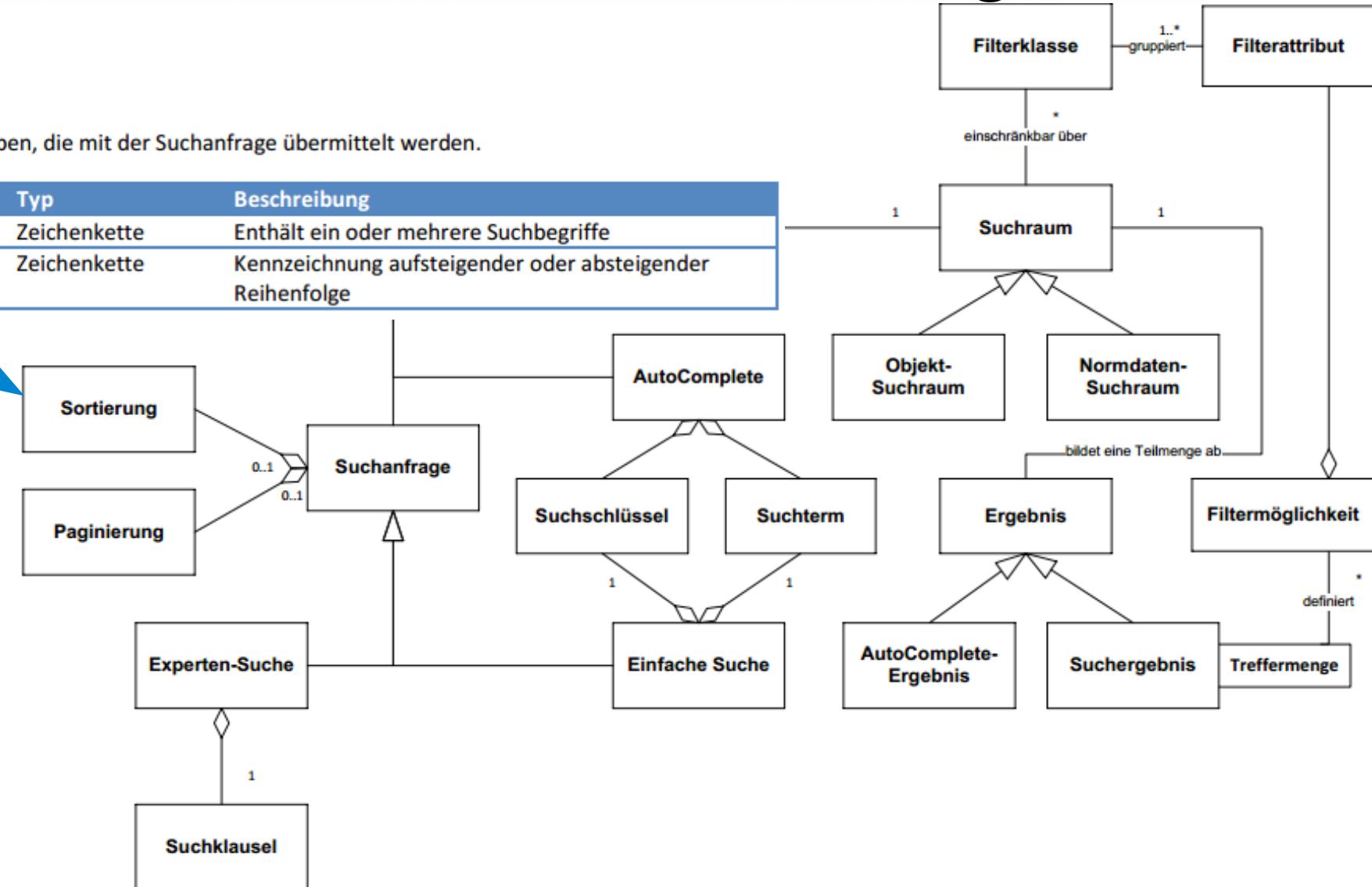


Abbildung 1: Fachmodell

Negationen vermeiden

4.1. Bedingungen für die Nicht-Indexierung eines Datensatzes

Jeder Pica-Plus-Datensatz wird zunächst auf die folgenden Merkmale hin untersucht:

Datensätze, die nicht den folgenden Mustern entsprechen, werden nicht indexiert

Falls die folgenden Bedingungen, entweder für Titel- oder für Normdaten, erfüllt werden, findet noch eine weitere Analyse (siehe nächste Tabelle) statt.

Datensatzart	Merkmal	Erläuterung
Titeldaten	002@\$0="^A-SU-Z).*"	Alle Datensätze, deren Inhalt in 002@\$0 mit einem Großbuchstaben beginnt, der nicht T entspricht
Normdaten	002@\$0="^T[bfgnpqsvu].* ^The ^T[kq]\$"	Alle Datensätze, deren Inhalt in 002@\$0 mit Tb,Tf,Tg,Tn,Tp,Tq,Ts,Tu,Tv beginnt oder The, Tq oder Tk entspricht

Positiv formulieren

Aus

„Datensätze, die nicht den folgenden Mustern entsprechen, werden nicht indexiert“

wird

„Nur Datensätze, die den folgenden Mustern entsprechen, werden indexiert“

Die Implementierung kann eigenständig zwischen einem Inklusions- oder Exklusionsfilter wählen

Tests aus fachlicher Spezifikation

3.1.2. Trunkierung

Suchbegriffe sollen links, rechts und in der Mitte trunkiert werden können. Folgende Zeichen sollen für die Trunkierung verwendet werden können:

Operator/ Zeichen	Beschreibung	Beispiel	PR-312: Trunkierung von Suchbegriffen (New Feature)
*	Platzhalter für <u>eine beliebige Anzahl</u> von Zeichen (0 oder mehr Zeichen)	<p>Suche: fähr*e Es werden Titel gefunden mit den Stichwörtern Fähre, Fährte, Fährbrücke etc.</p> <p>Suche: politisch*</p> <p>Es werden Titel gefunden mit den Stichwörtern politisch, politische, politischer etc.</p> <p>Suche: *wahrnehmung</p> <p>Es werden Titel gefunden mit den Stichwörtern Zeitwahrnehmung, Selbstwahrnehmung, etc.</p> <p>Suche: Ka*a*stan</p> <p>Es werden Titel gefunden mit den Stichwörtern Kasachstan, Kasachstans, Kasakhstan, Kazachstan, Kazakhstan, etc.</p>	<p>(Unit?) Test #1</p> <p>(Unit?) Test #2</p> <p>(Unit?) Test #3</p> <p>(Unit?) Test #4</p>

Ein Beispielhafter Test

Suche: politisch*

Es werden Titel gefunden mit den Stichwörtern politisch, politische, politischer etc.

```
public class TruncationTest {
```

```
    /**
     * #Requirement https://jira.dhbw.de/browse/PR-312
     */
```

```
@Test
```

```
public void rightSidedTruncation() {
    final Searchterm term = new Searchterm("politisch*");
    final Iterable<Book> found = new Search().of(term);
    assertThat(
        found,
        everyTitleContainsAnyOf(
            "politisch",
            "politische",
            "politischer",
            "politischem")));
}
```

PR-312: Trunkierung von Suchbegriffen (New Feature)

Ein Beispielhafter Test mit Mock

```
public class TruncationTest {  
  
    /**  
     * #Requirement https://jira.dhbw.de/browse/PR-312  
     */  
    @Test  
    public void rightSidedTruncation() {  
        final Searchterm term = new Searchterm("politisch*");  
        final Search search = mockedSearch().with(  
            "Das politisch interessierte Jahrbuch",  
            "Der politische Programmierer",  
            "Ein politischer Fehlritt. Das Leben des ...",  
            "Mit politischem Gespür. Biographie von ...",  
            "Ganz diplomatischer Buchtitel");  
        final Iterable<Book> found = search.of(term);  
        assertThat(found,  
            everyTitleContainsAnyOf(  
                "politisch",  
                "politische",  
                "politischer",  
                "politischem")));  
    }  
}
```

Mocking:

Testtechnik,
bei dem
echte
Funktionalität
durch
vorbereitete
Ergebnisse
ersetzt wird

Weiteres Beispiel

2.3.1.2 Existenz eines Namensraums im System abfragen

Es kann auch die Existenz eines Namensraums im System über dessen Namen abgefragt werden. Der Name wird in der URL angegeben.

2.3.1.2.1 Anfrage

HEAD /namespaces/name/ :name



URN-231: Namensraum-existenz abfragen

2.3.1.2.2 HTTP-Response-Status

Status	Bedeutung
200 OK	Es wurde eine korrekte Anfrage empfangen, die fehlerfrei behandelt werden konnte.
401 Unauthorized	Der Benutzer konnten nicht authentifiziert werden (Fehlercode: 401001)
404 Not Found	Der Namensraum mit diesem Namen existiert nicht im System (Fehlercode: 404001)

2.3.1.2.3 Response-Body

Im Fehlerfall wird ein *ErrorResponse*-Objekt (siehe 2.1.11.1) geliefert.

2.3.1.2.4 Beispiel

Request
Header:
Host: example.org

Response
Header:
Status: 200 OK

(Unit?) Test #1

Umsetzung hier: Integrationstest

2.3.1.2.4 Beispiel

Request HEAD /v1/namespaces/name/urn%3Anbn%3Ade%3A1111

Header: Host: example.org

Response Status: 200 OK

Header:

URN-231: Namensraum-existenz abfragen

```
public class NamespaceExistenceQueryTest {  
  
    /**  
     * #Requirement https://jira.dhbw.de/browse/URN-231  
     */  
  
    @Test  
    public void namespaceExists() {  
        final WebConversation conversation = new WebConversation();  
        final WebRequest request = new HeadMethodWebRequest(  
            "http://example.org/v1/namespaces/name/urn%3Anbn%3Ade%3A1111");  
        final WebResponse response = conversation.getResponse(request);  
  
        assertThat(response.getResponseCode(), equalTo(200));  
        assertThat(response.getHeaderFieldNames(), emptyArray());  
    }  
}
```

Umsetzung hier: Integrationstest

```
public class NamespaceExistenceQueryTest {  
  
    /**  
     * #Requirement https://jira.dhbw.de/browse/URN-231  
     */  
    @Test  
    public void namespaceExists() {  
        final String urnServerAddress = "http://localhost:8080";  
        startURNServerAt(urnServerAddress);  
  
        final WebConversation conversation = new WebConversation();  
        final WebRequest request = new HeadMethodWebRequest(  
            urnServerAddress + "/v1/namespaces/name/urn%3Anbn%3Ade%3A1111");  
        final WebResponse response = conversation.getResponse(request);  
  
        assertThat(response.getResponseBody(), equalTo("urn:urn%3Anbn%3Ade%3A1111"));  
        assertThat(response.getHeaderFieldNames(), emptyArray());  
    }  
}
```

Testen nicht-funktionaler Anforderungen

Qualitätsanforderungen an die PDF-Extraktion

Die Anzahl der fehlerhaften Texte darf 5% nicht überschreiten.

Ein Text gilt als fehlerfrei, wenn mindestens 95% der Wörter korrekt sind.

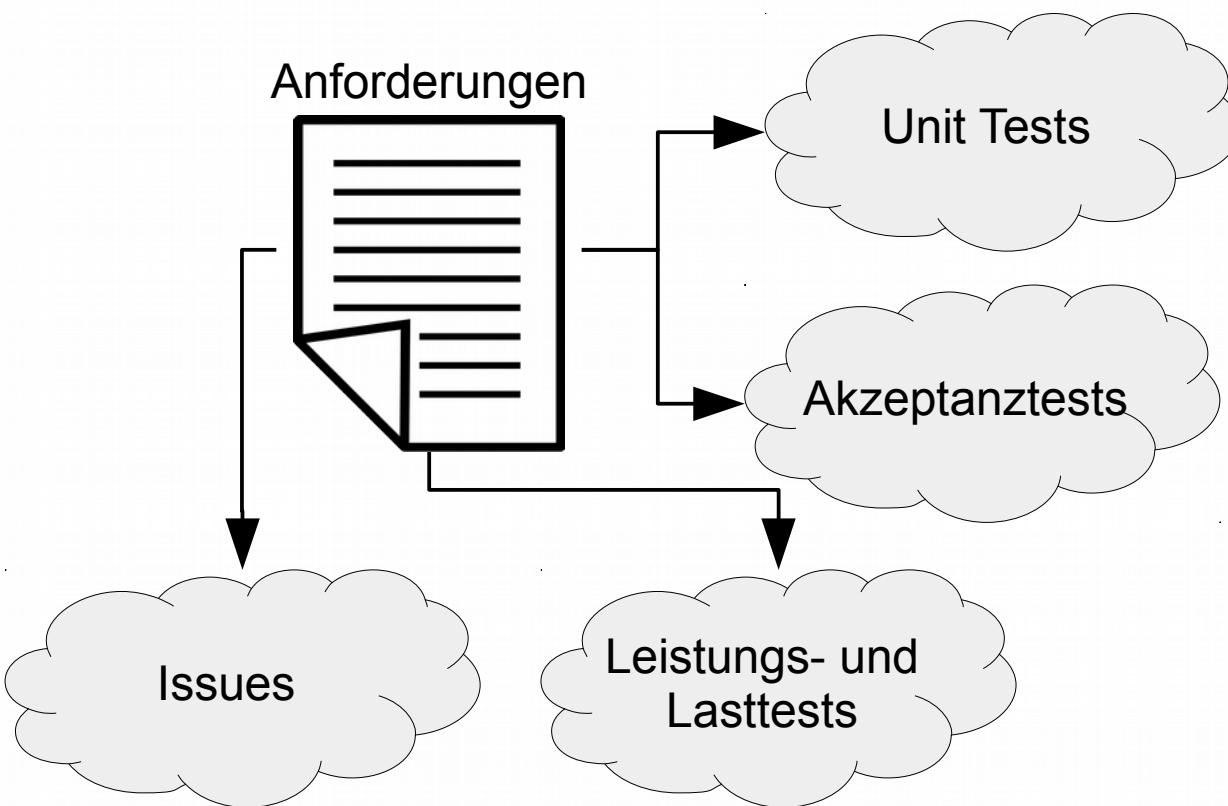
Qualitätsanforderungen an die EPUB-Extraktion

Die Anzahl der fehlerhaften Texte darf 3% nicht überschreiten.

Ein Text gilt als fehlerfrei, wenn mindestens 98% der Terme korrekt sind.

- Zwei verschiedene Angaben
 - Fehlerfreier Text = 95/98 % Wörter/Terme korrekt
 - Anteil fehlerfreier Texte größer/gleich 95/97 %
- Erste Angabe ist eine Definition → Unit Test
- Zweite Angabe benötigt eine andere Art Test

Nachweismittel von Anforderungen



- Funktionale A.
 - Unit Test
- Nicht-funktionale A.
 - Leistungstest
 - Lasttest
- Fachliche A.
 - Akzeptanztest
- Technische A.
 - Integrationstest

Testen nicht-funktionaler Anford.

- „Anteil fehlerfreier Texte größer/gleich 95/97 %“
- Erfordert Leistungstest
- Repräsentative Menge an Texten
- Texte werden (meist manuell) ausgewertet
 - Referenzergebnis
- Prüfen der Algorithmusgüte gegen das Referenzergebnis
- Möglichst bei Projektbeginn schon aufsetzen
 - Dient als Abbruchkriterium für Algorithmusentwicklung

Qualität des Anforderungsdokuments

3.6. Datenabfrage im URN-Service

- Jede Entität kann bei Kenntnis ihres eindeutigen Bezeichners (Unternamensraumpräfix, URN, Login, Institutions-Kennung) direkt adressiert und abgerufen werden. Bei URLs ist die Kombination aus URL und URN der eindeutige Identifikator
- Alle Entitäten können jeweils als komplette Liste abgerufen werden (wenn die Menge dies zulässt)
- Zusätzlich ist es möglich, bestimmte Teilmengen der Entitäten abzurufen:
 - Unternamensräume pro Institution
 - URLs pro Institution
 - Benutzer pro Institution
 - URNs pro Unternamensraum
 - URNs pro Erstellungs-/Änderungsdatum
 - URLs pro URN
 - URLs pro Institution pro URN
 - URLs pro Erstellungs-/Änderungsdatum
 - Institution eines Benutzers

Kommentar [cb70]: Wie soll damit umgegangen werden, wenn die Menge zu groß ist? Soll dann seitenweise geblättert werden?

Kommentar [cb71]: Geht es hier um eine Statistik oder um die konkrete Auflistung aller URNs die an einem Tag X erstellt/geändert wurden?
Soll dies eine kombinierte Abfrage sein (also erstellt oder geändert) oder sind es zwei einzelne Anfragen?

Kommentar [cb72]: Siehe Kommentar [cb50]

Kommentar [cb73]: Ist das so zu verstehen, dass die URN-REST-API eine Möglichkeit bieten muss, dass Administratoren in SQL formulierte Anfragen ausführen können sollen oder soll ein von der URN-REST-API unabhängiger direkter Zugriff auf die Datenbasis möglich sein?

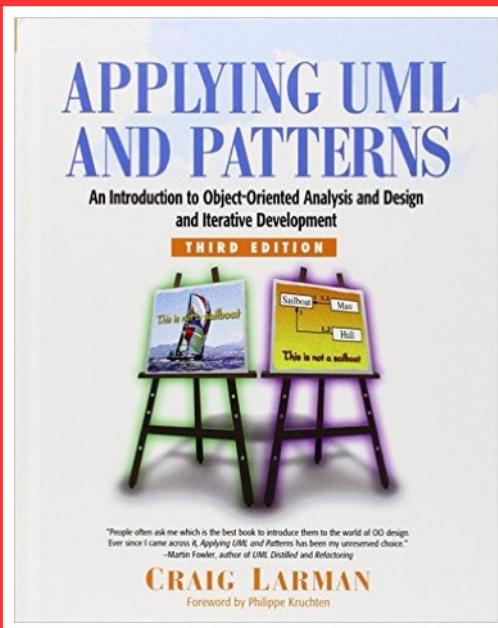
Wenn die erste Variante angedacht ist, dann ist eine genaue Beschreibung des zu unterstützenden Sprachumfangs notwendig. Ich bitte außerdem zu bedenken, dass es implementierungstechnisch ziemlich aufwendig ist die Anfragen zu

- Ausschnitt aus Seite 25 der fachlichen Spezifikation
- ca. 3 Fragen pro Seite → sofort weiteres Interview

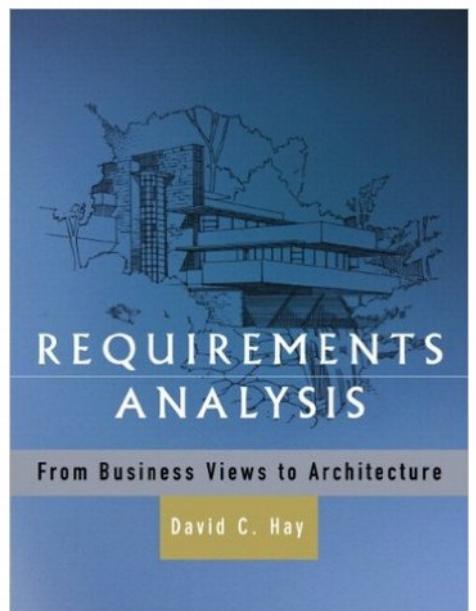
Qualität des Anforderungsdokuments

- Eure Einschätzung ist wichtig und richtig
 - Anforderungsdokumente werden für den Leser geschrieben
 - Wenn ihr es nicht „versteht“, ist es nicht gut
- Anforderungsdokumente zu schreiben ist sehr schwierig (und aufwendig)
 - Viel Geduld und Mithilfe notwendig
 - Schneide-Sprech: „Den Kunden entwickeln“
- Wichtig: Bewusster Perspektivwechsel
 - Entwickler denken „von der anderen Richtung her“

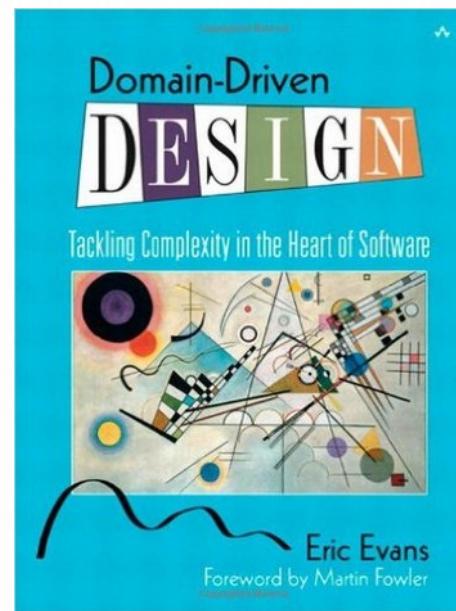
Anforderungsanalyse: Weiterführende Literatur



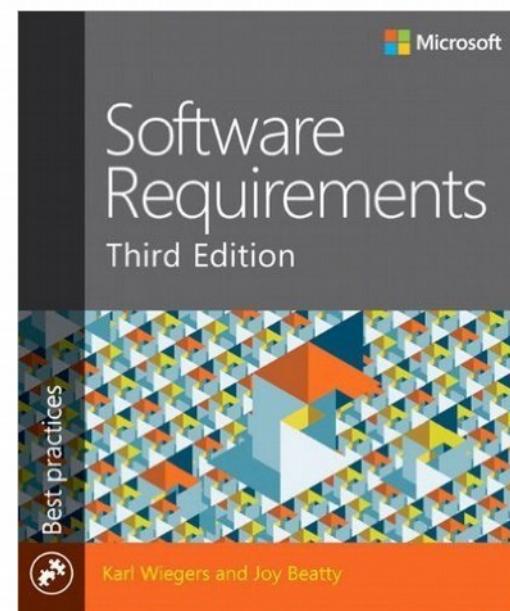
2004



2002



2003



2013

(2003)

(1999)