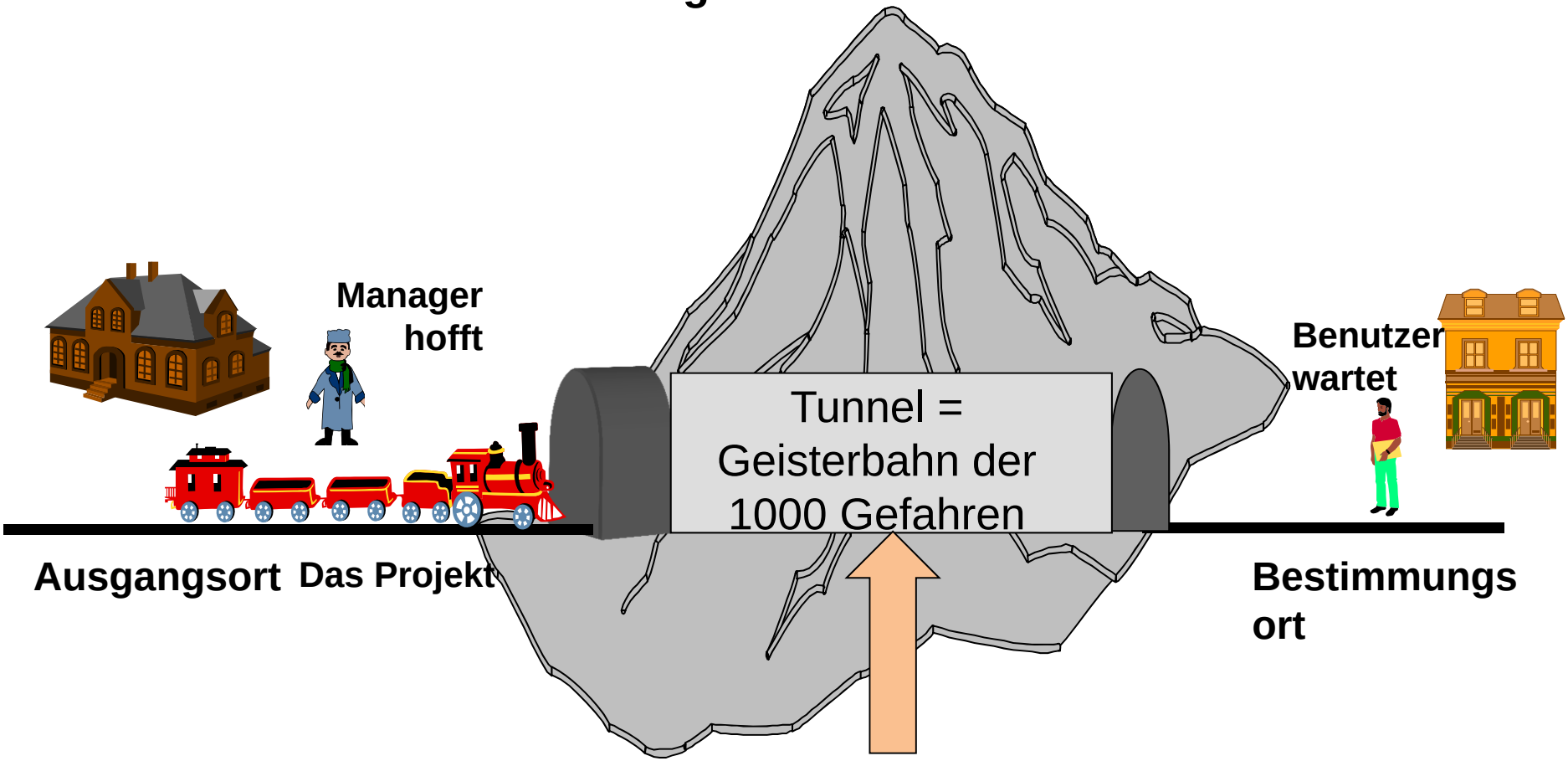
Tracing Software Systemevolution

Harry M. Sneed
SoRing Kft. H-1221 Budapest
Birgit Demuth
Technische Universität Dresden,
Institut für Softwaretechnik

GI Design for Future Workshop
Bad Honneff im Mai 2018

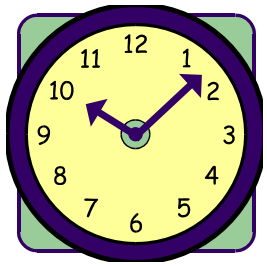
Die Ortung eines Softwareentwicklungsprojektes

Ist gar nicht so einfach, vor allem
wenn die wichtigsten Informationen fehlen



Wo steckt das Projekt?

Tracing a Development Project



Time is running

Summit
(6. Hours?)

Steep Cliffs

Here the next mountain is waiting

Glacier

3. MileStone
(4 Hours)

2. MileStone
(3 Hours)

Rock Slide

Follow Me!!!

Plan

1. MileStone
(1 Hour)

An Unknown Mountain

Project Team has only so much energy

Projects are a one time restricted effort to reach a predefined Goal

Finding Where a Project is by reviewing the Results

Es hat keinen Sinn mit den Entwicklern über den Stand ihrer Arbeit zu reden. Sie werden die Wahrheit immer verschleiern.
Sie wissen selber nicht wo sie stehen.
Insofern führt der interview Ansatz ins Leere. Nur die Ergebnisse zählen – only the results count.

Software-Engineering	**➔**	**Order**
Order	**➔**	**Rules**
Rules	**➔**	**Discipline**
Discipline	**➔**	**Control**

„It is not that software developers do not believe in quality, it is that under pressure they can not resist the temptation to cheat.
They are simple sinners and like all sinners they respond best to the threat of punishment“

(John Shore: Confessions of a lapsed Software Engineer or why I never met a programmer I could trust)

Tracing the Course of a Software Project

Forward Tracing

Requirement Docu = Lastenheft

Func_Requ_2: Accept Orders from authorized Customers

Subject Model

Functional Spec = Pflichtenheft System Design = Objectmodel

UC: Accept_Order
Pre: Items on Stock
Post: Items = #Items - Order

Order	Cust omer	Item	Bill
Place →	Check	Check	
	→	Deduct	
	→		Charge

Code

Class: Item
Method: Check(OnStock)
Method: Deduct(Order)
Method Get (Price)

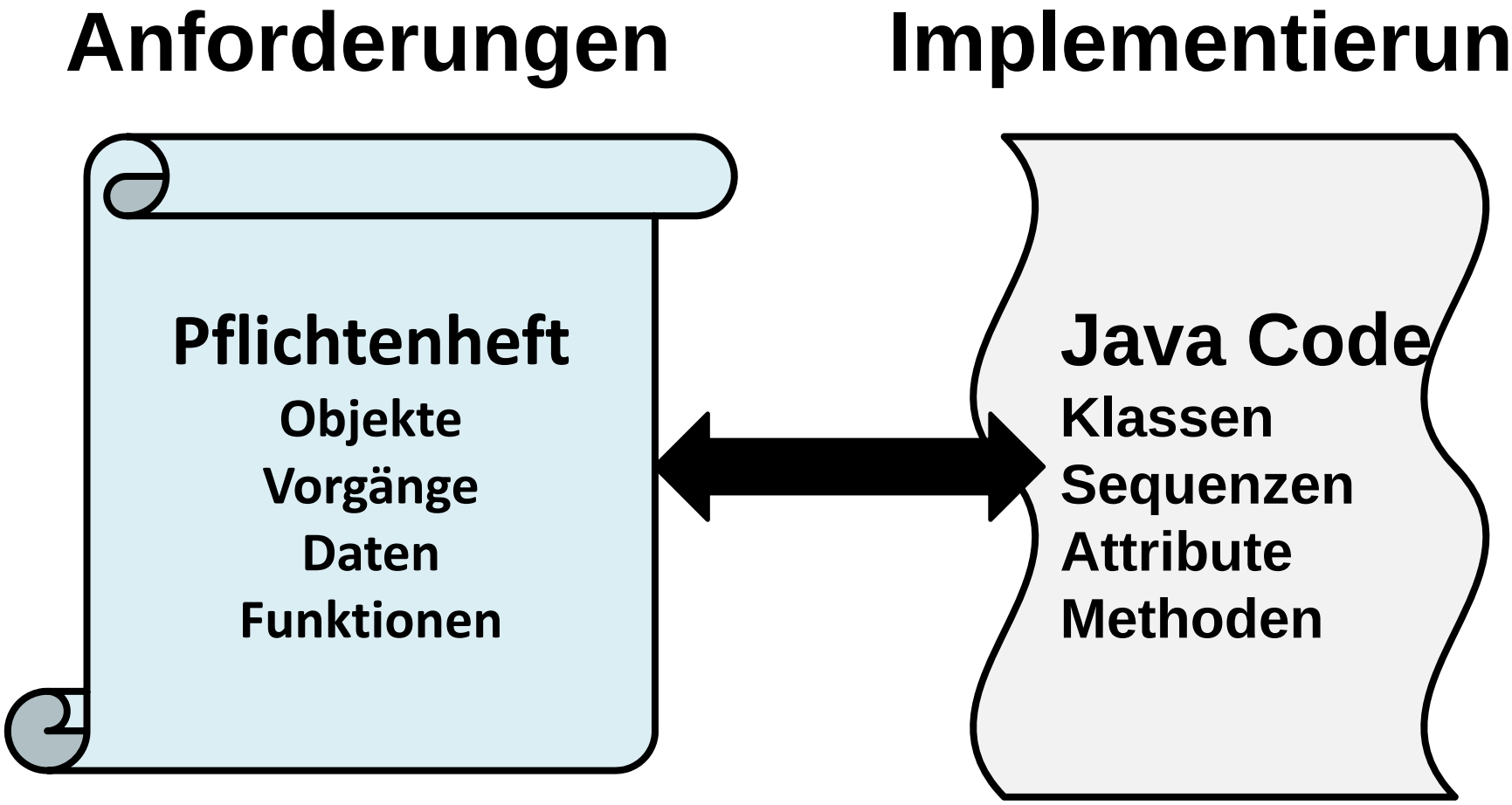
Test Cases

TC_1: Place Order for Unknown Customer
TC_2: Place Order on missing item
TC_3: Place Order on insufficient OnStock :

Requirement is specified in UseCase
Use Case is designed (modeled)
Use Case is coded
Use case is tested

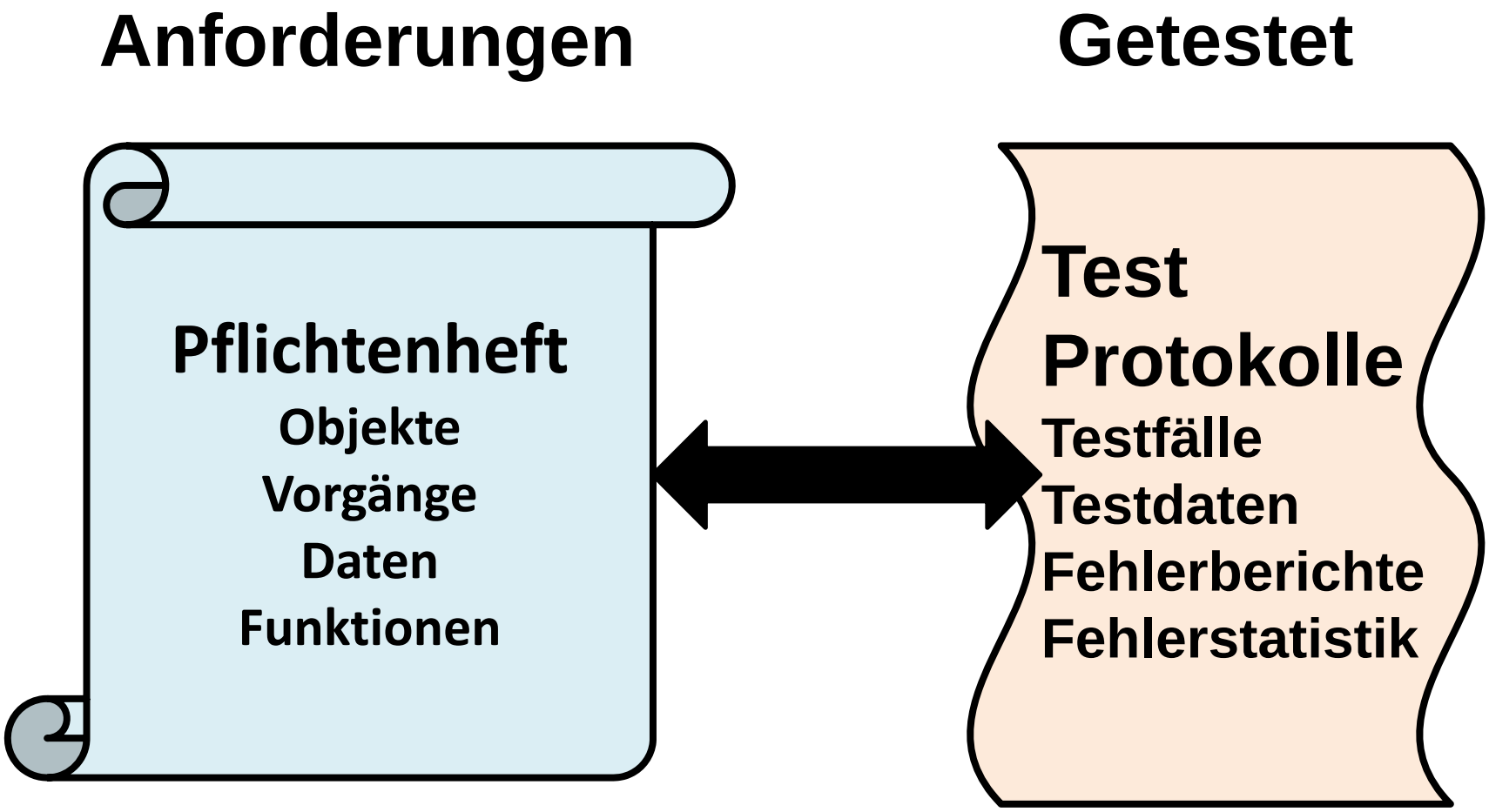
Backward Tracing

Abgleich des Codes mit den Anforderungen



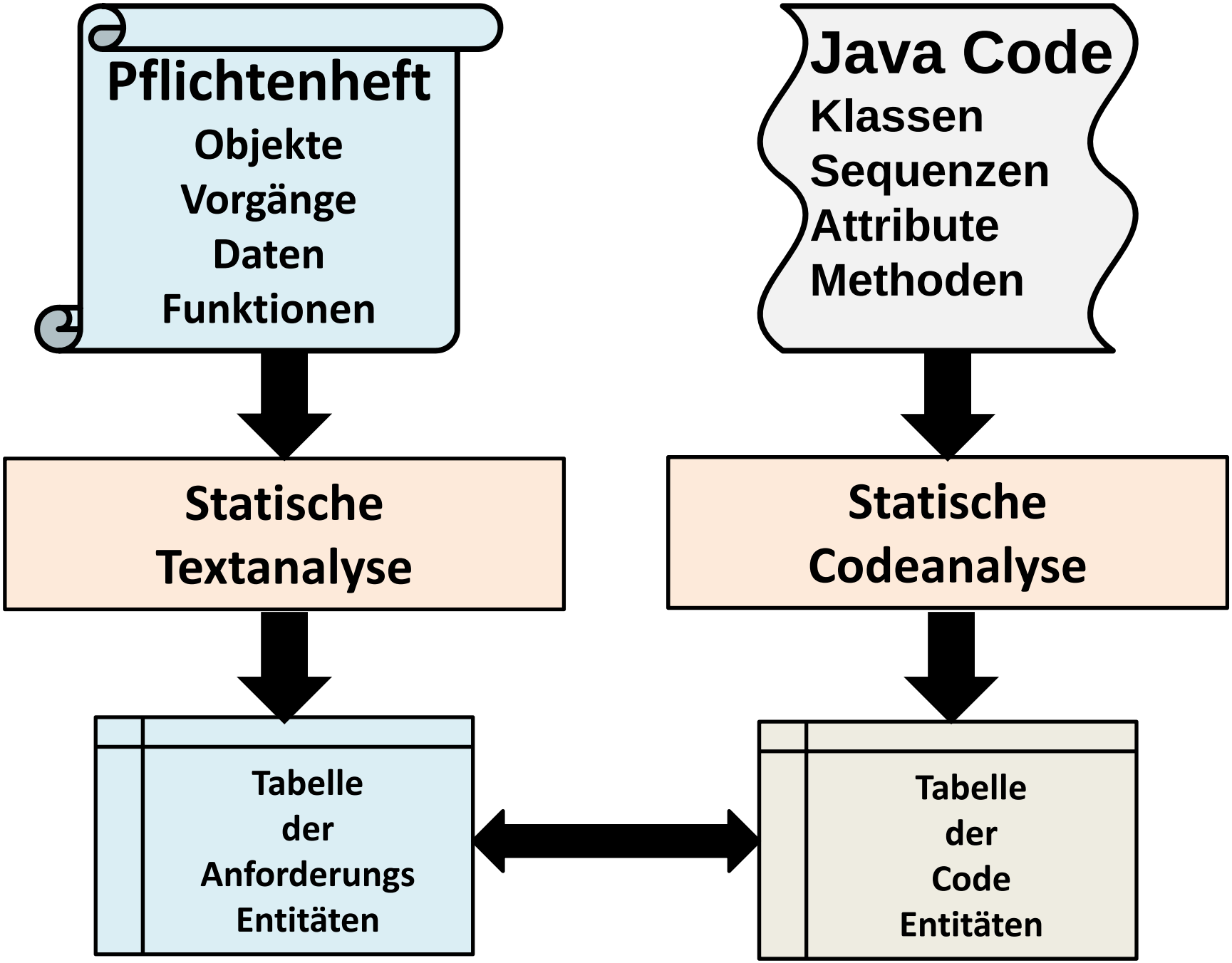
Wieviel von den angeforderten Daten und Funktionen sind in dem Code bereits vorhanden.

Vergleich Anforderungen mit getesteten Testfälle



Wieviel von den angeforderten Daten und Funktionen Sind bereits getestet ?

Parallele Untersuchung vom Dokument und Code



Requirement	Use Case	Design	Code	Test	Test Cases
REQ-1: Place Order	UC-A	Diagram-A	Module-A1	TEST-A	TC-A1, TC-A2, TC-A3
		Diagram-A1	Module-A2		
REQ-2: Check Order	UC-A	Diagram-A	Module-A3		
			Module-X1		
REQ-3: Check Customer	UC-A	Diagram-A	Module-A4		
		Diagram-A2	Module-X2		
REQ-4: Process Order	UC-B	Diagram-B	Module-B1	TEST-B	TC-B1, TC-B2, TC-B3, TC-B4, TC-B5
		Diagram-B1	Module-B2		
		Diagram-B2	Module-B3		
			Module-X3		
REQ-5: Create Bill	UC-C	Diagram-C	Module-C1	TEST-C	TC-C1, TC-C2,TC-C3, TC-C4
		Diagram-C1	Module-C2		
			Module-X4		
REQ-6: Update Sales	UC-C	Diagram-C1	Module-C3		
		Diagram-C2	Module-C4		
REQ-7: Replenish Stock	UC-D	Diagram-D	Module-D	TEST-D	TC-D1, TC-D2, TC-D3

Fragen zum Stand eines Projektes

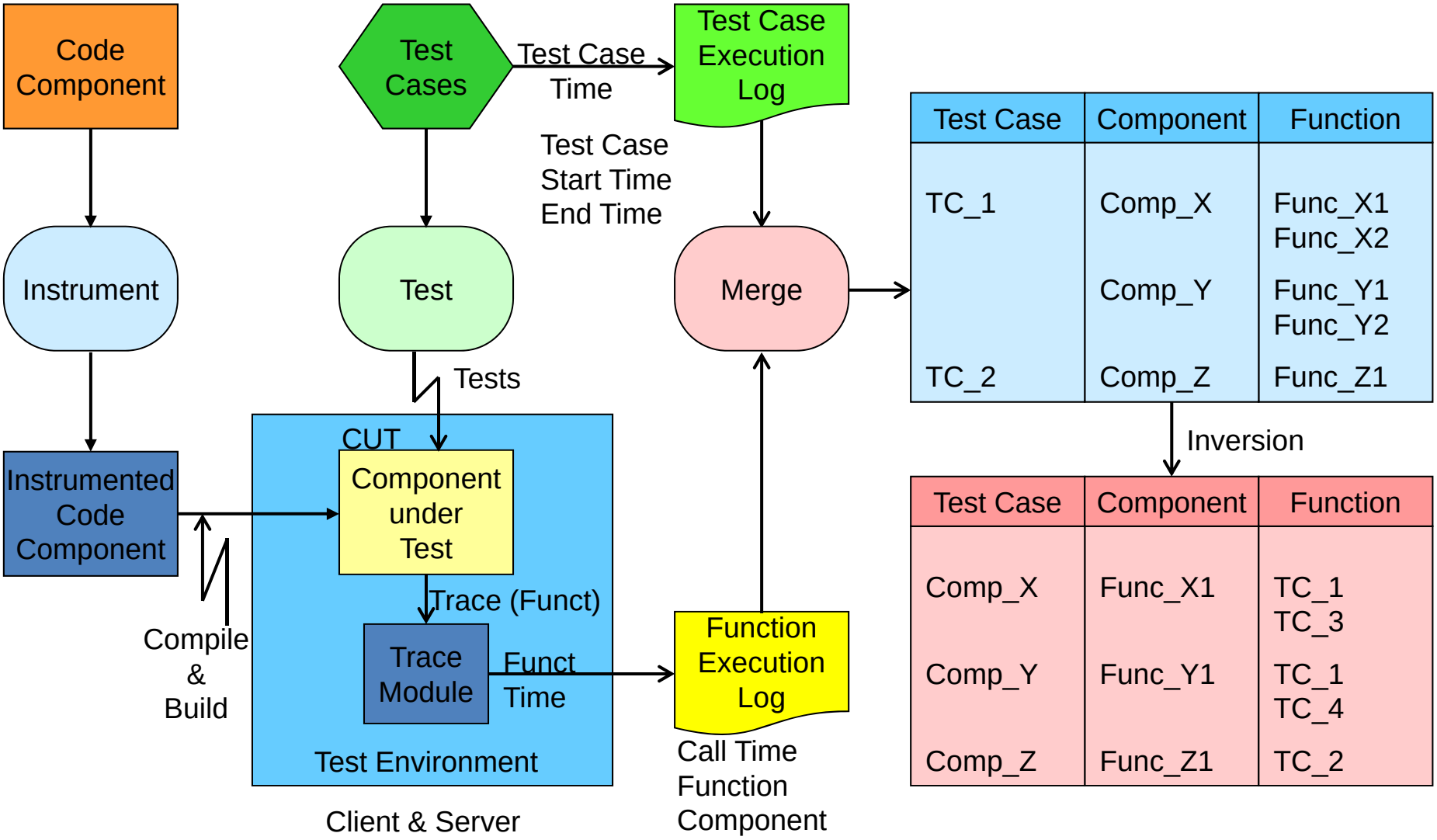
- **Frage: Wie können wir wissen ob eine Anforderung erfüllt ist oder nicht und wenn ja von welchem Code?**
- **Frage: Wie können wir wissen ob eine Anforderung getestet wurde und wenn ja von welchem Testfall?**
- **Mögliche Antworten**
  - **Dynamische Analyse = Teste jede spezifizierte Anforderung mit allen Testfällen und verfolge den Ablauf der Testfälle durch den Code**
    - **Teuerste Antwort**
    - **Setzt eine Testumgebung und geeignete Testdaten voraus**
    - **Ist sehr aufwendig**
  - **Statische Analyse = Vergleiche den Source Code und die Testfallbescreibungen mit der Anforderungsspezifikation (Feature Analysis)**
    - **Billigste Antwort**
    - **Setzt Analysewerkzeuge und Namenskonvention voraus**
    - **Ist nicht so genau**

Projektortung eines Smart-Metering Projektes

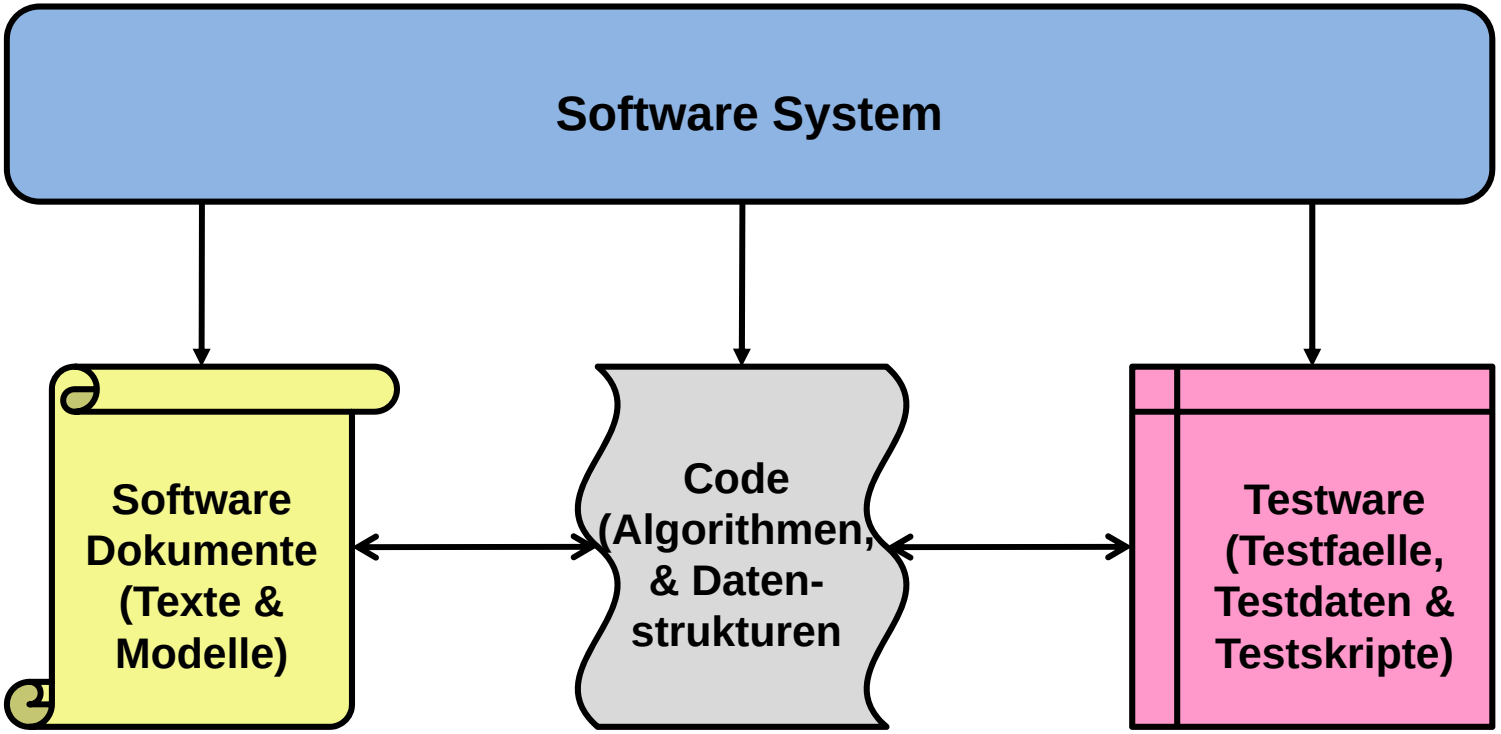
- **Eine österreichische Stromversorgungs-gesellschaft hatte einen Auftrag erteilt an ein belgisches Softwarehaus ein Smart Metering System zu entwickeln.**
- **Der Test des Systems sollte ein anderes Softwarehaus in Rumänien durchfuhren.**
- **Mitten im Projekt wurde das belgisches Softwarehaus verkauft und hat sich von dem Projekt verabschiedet. Das rumänische Softwarehaus sollte die Entwicklung übernehmen.**
- **Ein Lastenheft mit 122 detaillierten Anforderungen, ein Pflichtenheft mit 87 Anwendungsfällen, ein Test mit 144 spezifizierten Testfällen und mehr als 800 Java Klassen lagen vor.**
- **Der Kunde will wissen wo das Projekt steht und wie viel muss er noch investieren um das Projekt abzuschließen.**
- **Du hast zwei Monate Zeit um die Frage zu beantworten.**
- **Wie gehst Du vor?**

Der dynamische Testansatz

Du könntest jede Anforderung testen um zu prüfen ob er erfüllt ist. Das wäre ein großer Aufwand und würde mehrere Monate dauern.



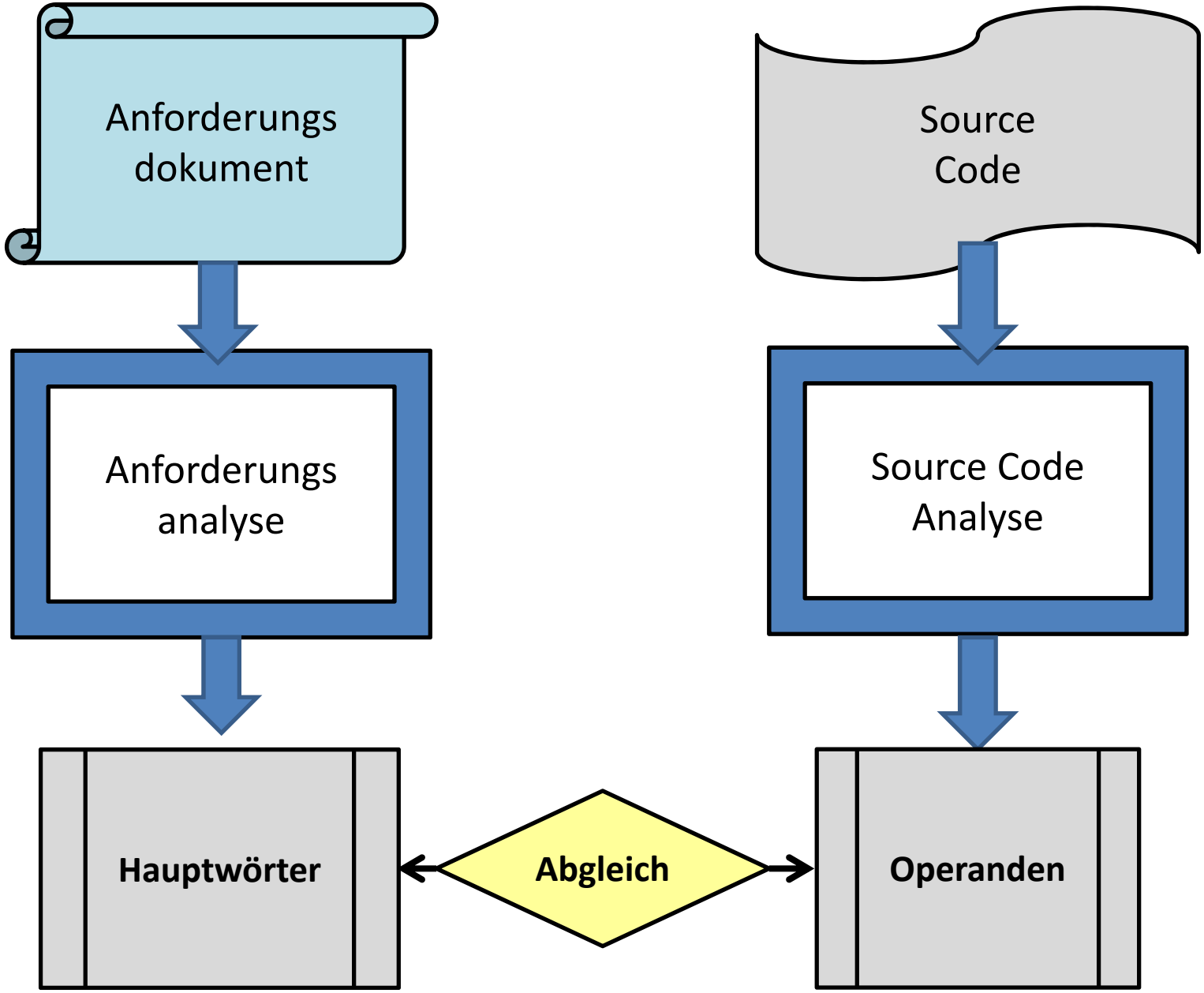
Der statische Feature Analyse Ansatz



Du verfolgst jede Anforderung durch die Dokumentation, den Code und die Testfälle um festzustellen wo sie überall vorkommt. Dies wird als Feature Analysis bezeichnet..

Some Feature like „Standortermittlung“ wird überall gesucht.

Abgleich der Datennamen



Verbindung über die Hauptwörter im Text

Wichtig sind die Substantive im Text, Wörter wie Buchhalter, Rechnungsstellung, Rechnungsposten und Kunde. Sie identifizieren die Objekte, die von dem Anwendungsfall verarbeitet werden. Zu jeder Anforderung und jedem Anwendungsfall werden diese Objekte gesammelt und in einem Repository gespeichert:

PROC;Auftragsbearbeitung ;OWNS;CASE; Rechnungsstellung
CASE;Rechnungsstellung ;HAS ;ACT ; Buchhalter
CASE;Rechnungsstellung ;HAS ;TRIG; Menuauswahl
CASE;Rechnungsstellung ;FILL; REQU; stelle monatliche Rechnung.
CASE;Rechnungsstellung ;USES;OBJT; GO-01-Kunde
CASE;Rechnungsstellung ;USES;OBJT; GO-06-Rechnungsposten
CASE;Rechnungsstellung ;INPT; MASK;GUI-Rechnungsmaske
CASE;Rechnungsstellung ;OUTP;REPO; LIST-05-Rechnung
CASE;Rechnungsstellung ;OUTP;REPO; LIST-06-Rechnungsprotokoll
CASE;Rechnungsstellung ;IMPL; RULE; GR-09-Rechnungsbetrag
CASE;Rechnungsstellung ;IMPL; RULE; GR-10-Mehrwert
CASE;Rechnungsstellung ;HAS ; PRE ; Rechnungsposten liegt vor
CASE;Rechnungsstellung ;HAS ; POST; Rechnungen sind gedruckt
CASE;Rechnungsstellung ;PERF; STEP; Buchhalter startet Rechnungsstellung.
CASE;Rechnungsstellung ;USES; DATA; Kundennr
CASE;Rechnungsstellung ;USES; DATA; Gesamtbetrag

&AN03.04.01 - Standort-medizinischeFachrichtung führen
Priorität High
Beschreibung

Hauptwörter = Standorte, Fachgebiete, Fachrichtungen

Für:
. alle nicht stornierten Standorte innerhalb einer Krankenanstalt in ZPV4.0, in denen Leistungen aus den **Fachgebieten** 080, 081, 084, 086, 091 erbracht werden und
. alle nicht stornierten Standorte innerhalb einer MSE in ZPV4.0, in denen Leistungen aus dem Fachgebiet 085, 087 oder 099 erbracht werden und
. alle nicht stornierten Standorte innerhalb einer Vorsorgeuntersuchungsstelle in einem Betrieb in ZPV4.0, in denen Leistungen aus dem **Fachgebie**t 098 erbracht werden und
. alle nicht stornierten Standorte zu Sammelpartnern in ZPV4.0 mit den Fachgebieten 080, 081, 085, 087, 098 oder 099
gilt:
ZPV4.0 muss zu jedem Standort alle zum Beobachtungszeitpunkt unaktuellen, aktuellen und stornierten medizinische Fachrichtungen führen, die die dort erbrachten Leistungen aus den oben genannten **Fachgebieten** näher beschreiben.

&AN03.04.02 - Standort-medizinischeFachrichtung abbilden
Priorität High
Beschreibung
ZPV4.0 muss jedem in AN03.04.01 angeführten Standort während dessen Gültigkeitszeitraum:
. mindestens eine oder mehrere aktuelle **Standort**-medizinischeFachrichtungen führen (Zum selben Zeitpunkt dürfen nur unterschiedliche **Standor**t-medizinischeFachrichtungen zum Standort aktuell sein)
. oder mehrere unaktuelle **Standor**t-medizinischeFachrichtungen führen (Zum selben Zeitpunkt dürfen nur unterschiedliche Standort-medizinischeFachrichtungen zum Standort unaktuell sein)
. oder mehrere stornierte **Standor**t-medizinischeFachrichtungen führen (Zum selben Zeitpunkt dürfen sowohl gleiche als auch unterschiedliche Standort-medizinischeFachrichtungen zum **Standor**t storniert sein).
Zwei Standort-medizinischeFachrichtungen eines Standorts gelten im Kontext dieser Anforderung als gleich, wenn sie:

Requirement Beziehungstabelle

Type;Base Entity	;Rel ;Type;Target Entity
PROD;HVB	;OWNS;SYST;ZPV4
SYST;HVB	;OWNS;DOCU;100_ZPV4.0
DOCU;100_ZPV4	;OWNS;REQU;&AN03.04.01
REQU;&AN03.04.01	;USES;DATA;Standort-Fachrichtung
REQU;&AN03.04.01	;USES;DATA;Standorte
REQU;&AN03.04.01	;USES;DATA;Krankenanstalt
REQU;&AN03.04.01	;USES;DATA;Fachgebieten
REQU;&AN03.04.01	;USES;DATA;Standorte
REQU;&AN03.04.01	;USES;DATA;Fachgebiet
REQU;&AN03.04.01	;USES;DATA;Vorsorgeuntersuchungsstelle
REQU;&AN03.04.01	;USES;DATA;Betrieb
REQU;&AN03.04.01	;USES;DATA;Fachgebiet
REQU;&AN03.04.01	;USES;DATA;Sammelpartnern
REQU;&AN03.04.01	;USES;DATA;Standort
DOCU;100_ZPV4	;OWNS;REQU;&AN03.04.02
REQU;&AN03.04.02	;USES;DATA;Standort-Fachrichtung
REQU;&AN03.04.02	;USES;DATA;Standort
REQU;&AN03.04.02	;USES;DATA;Gueltigkeitszeitraum
REQU;&AN03.04.02	;USES;DATA;Standort

&BAF04.03 - Liste der Standortfachgebiete anzeigen
Kurzbeschreibung
Dieser Anwendungsfall beschreibt die Anzeige einer Liste von **Standortfachgebieten** zu einer **Standort-Klassifizierung**.
ArtderDurchführung online
&Akteur allg-ZPV-Benutzer

Namen = Standort, Fachgebiet, Klassifizierung

Vorbedingungen
. **Standortfachgebiet** ist in ZPV4.0 vorhanden.
. **Standortfachgebie**t ist durch Benutzer identifiziert.
Endzustand im Gutfall
. Liste der **Standortfachgebiete** zu einer **Standor**t-**Klassifizierung** ist in ZPV4.0-Benutzeroberfläche angezeigt.
Endzustand im Fehlerfall
. Fehlermeldung ist in einem neuen Fenster ausgegeben.

&BAF04.04 - einzelnes Standortfachgebiet anzeigen
Kurzbeschreibung
Dieser Anwendungsfall beschreibt die Anzeige eines einzelnen Standortfachgebietes.
ArtderDurchführung online
&Akteur allg-ZPV-Benutzer
Vorbedingungen
. Das anzuzeigende **Standortfachgebie**t ist in ZPV4.0 vorhanden.
. Das anzuzeigende **Standortfachgebie**t ist durch Benutzer identifiziert.
Endzustand im Gutfall
. Das Standortfachgebiet wird mit seinen Attributen an der ZPV4.0-Benutzeroberfläche angezeigt.
Endzustand im Fehlerfall
. Fehlermeldung ist in einem neuen Fenster ausgegeben.

Use Case Beziehungstabelle

DOCU;103_ZPV4	;OWNS;CASE;&BAF04.03
CASE;&BAF04.03	;USES;DATA;Anwendungsfall
CASE;&BAF04.03	;USES;DATA;Anzeige
CASE;&BAF04.03	;USES;DATA;Liste
CASE;&BAF04.03	;USES;DATA;Standortfachgebieten
CASE;&BAF04.03	;USES;DATA;Standort-Klassifizierung
CASE;&BAF04.03	;HAS ;PRE ;. Standortfachgebiet ist vorhanden.
CASE;&BAF04.03	;USES;DATA;Standortfachgebiet
CASE;&BAF04.03	;USES;DATA;Benutzer
CASE;&BAF04.03	;HAS ;POST;im Gutfall
CASE;&BAF04.03	;USES;DATA;Liste
CASE;&BAF04.03	;USES;DATA;Standortfachgebiete
CASE;&BAF04.03	;USES;DATA;Standort-Klassifizierung
CASE;&BAF04.03	;HAS ;POST;im Fehlerfall
CASE;&BAF04.03	;USES;DATA;Fehlerfall
CASE;&BAF04.03	;USES;DATA;Fehlermeldung
CASE;&BAF04.03	;USES;DATA;Fenster

Java Code zur Standortklassifizierung

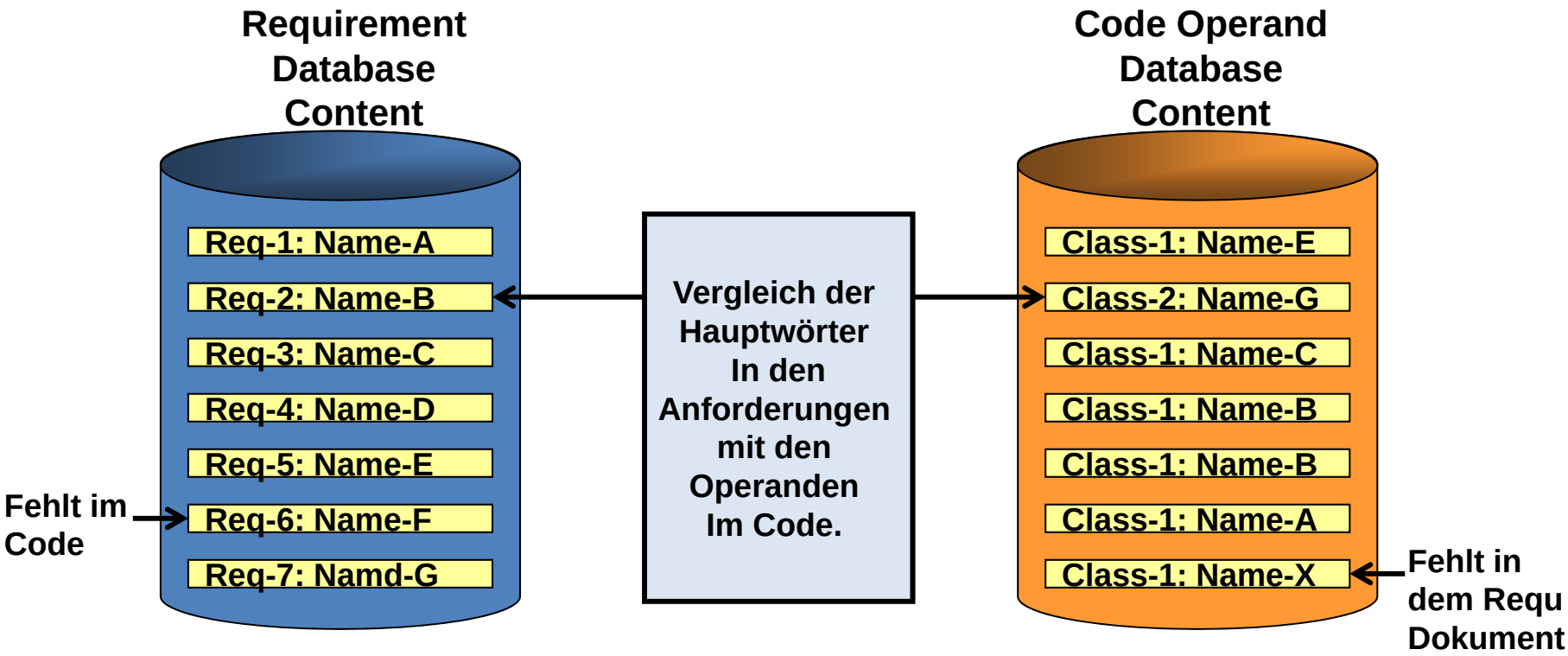

```
public IBERwStandortklassTypKlassFachg regelFuerStaKlassTypLEKlassTyp (
        java.lang.String standortklassifizierungTyp,
        java.lang.String leKlassifizierungKurz){
   if (standortklassifizierungTyp != null && leKlassifizierungKurz != null ) {
        for (IBERwStandortklassTypKlassFachg beRw : beRwList) {
            if (standortklassifizierungTyp.equals(beRw.getBeRwStandortklassifizierungTyp().getCode())
            && leKlassifizierungKurz.equals(beRw.getBeRwKlassifizierungFachgebiet().getBeRwKlassifizierungLE().
             getKlassifizierungLEKurz())) {
                return beRw;
             }
            }
             return null;
        }
  }
  public IBERwStandortklassTypKlassFachg regelFuerStandortklassTypKlassFachg (
            String standortklassifizierungTypKurz,
            String klassifizierungLEKurz,
            String fachgebietcode){
     if (standortklassifizierungTypKurz != null && klassifizierungLEKurz != null && fachgebietcode != null ) {
         for (IBERwStandortklassTypKlassFachg beRw : beRwList) {
            if (standortklassifizierungTypKurz.equals(beRw.getBeRwStandortklassifizierungTyp().getCode())
            && klassifizierungLEKurz.equals(beRw.getBeRwKlassifizierungFachgebiet().getBeRwKlassifizierungLE().
              getKlassifizierungLEKurz())
            && fachgebietcode.equals(beRw.getBeRwKlassifizierungFachgebiet().getBeRwFachgebiet().getFachgebietcode())
            && beRw.istGueltig()) {
              return beRw;
            }
     }
    return null;
  }
```

Operanden = Standort, Fachgebiet, Klassifizierung

Java Source Beziehungstabelle

/* ;DataDeclarations ;
SRC ;StandortklassTypKlassFachg;OWNS;FUNC;@Component
SRC ;StandortklassTypKlassFachg;OWNS;FUNC;regelFuerStaKlassTypLEKlass
SRC ;StandortklassTypKlassFachg; ;OWNS;FUNC;
regelFuerStandortklassTypKlassFachg
SRC ;StandortklassTypKlassFachg;OWNS;FUNC;load
SRC ;StandortklassTypKlassFachg;OWNS;FUNC;ist
GueltigeStandortklassTypKlassFachgKombination
SRC ;StandortklassTypKlassFachg;OWNS;FUNC;
listeAllerKlassFachgebieteZuStandort
SRC ;StandortklassTypKlassFachg;OWNS;FUNC;
listeDerMindestensEinKlassFachgebieteZuStandort
FUNC;listeDerMindestensEinKlassFachgebiet;OWNS;DATA;regel;regel
DATA;regel ;USES;TYPE;regel
SRC ;StandortklassTypKlassFachg;OWNS;FUNC;
listeDerGenauEinKlassFachgebieteZuStandort
FUNC;listeDerGenauEinKlassFachgebieteZuSt;OWNS;DATA;regel;regel
DATA;regel ;USES;TYPE;regel
SRC ;StandortklassTypKlassFachg.j
;OWNS;FUNC;listeDerZusaetzlichenKlassFachgebieteZuStandort
FUNC;listeDerZusaetzlichenKlassFachgebiet;OWNS;DATA;regel;regel
DATA;regel ;USES;TYPE;regel

Namenstabellenabgleich



Matching Names in a Requirement with Operands in the Code to locate the Class where the Requirement is fulfilled.

Anforderung **Linking Semantic Levels**

AN03.03.20 - Standort-Fachgebiete anzeigen
Priorität High
Beschreibung
ZPV4.0 muss dem Benutzer die Möglichkeit bieten, zu jedem Standort eines Leistungserbringers alle aktuellen, unaktuellen und stornierten **Standortfachgebiete** über die ZPV4.0-Benutzeroberfläche anzuzeigen.

Dabei muss ZPV4.0 dem Benutzer alle in AN03.03.02 und in den dazu untergeordneten Anforderungen definierten Eigenschaften der jeweiligen **Standortfachgebiete** anzeigen.

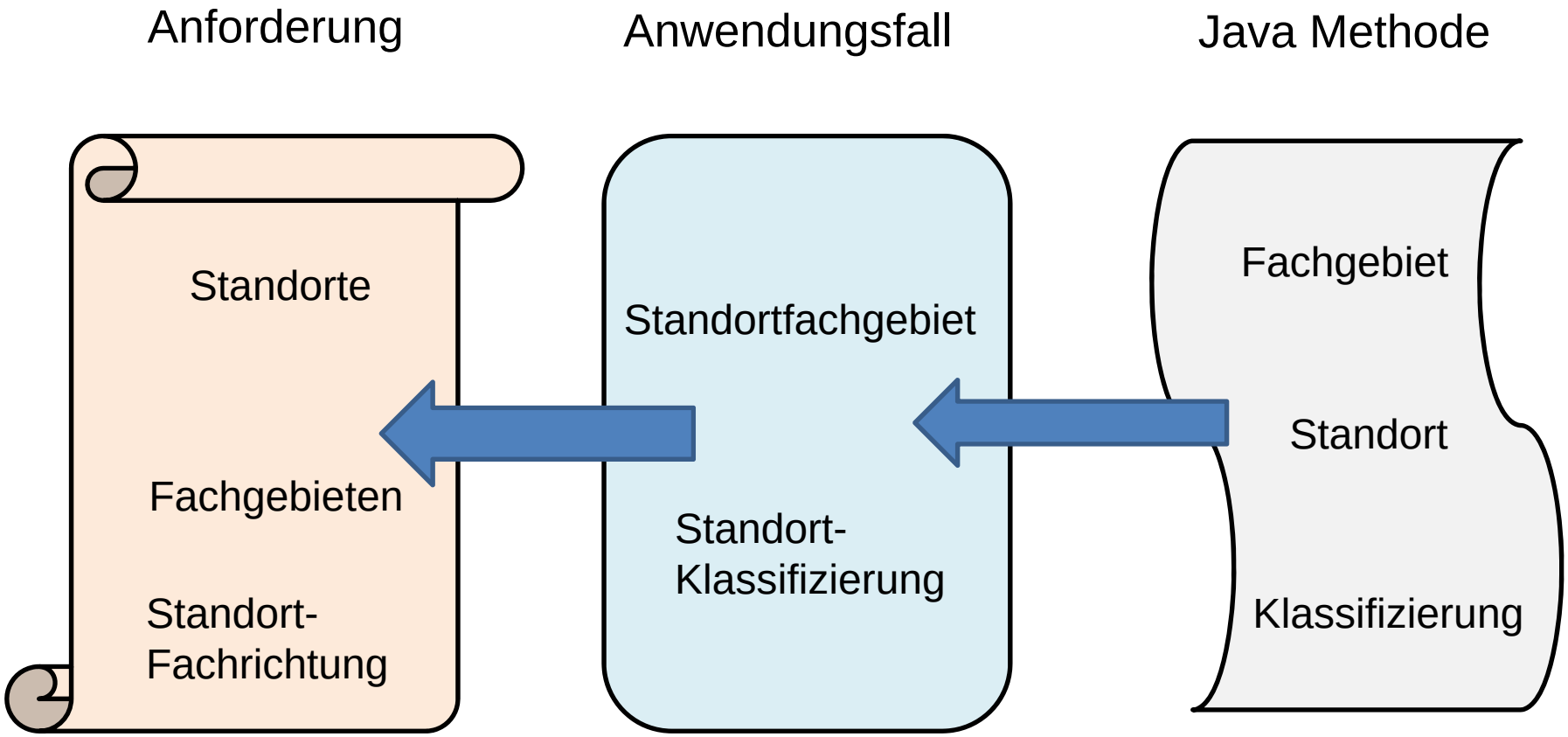
Anwendungsfall

&BAF04.04 - einzelnes Standortfachgebiet anzeigen
Kurzbeschreibung
Dieser Anwendungsfall beschreibt die Anzeige eines einzelnen **Standortfachgebiete**s.
ArtderDurchführung online
&Akteur allg-ZPV-Benutzer
Vorbedingungen
. Das anzuzeigende **Standortfachgebiet** ist in ZPV4.0 vorhanden.
. Das anzuzeigende **Standortfachgebiet** ist durch Benutzer identifiziert.
Endzustand im Gutfall
. Das **Standortfachgebiet** wird mit seinen Attributen an der ZPV4.0-Benutzeroberfläche angezeigt.

Java Code

```
if (standortklassifizierungTypKurz.equals (beRw.getBeRwStandortklassifizierungTyp().getCode()) &&
klassifizierungLEKurz.equals(beRw.getBeRwKlassifizierungFachgebiet().getBeRwKlassifizierungLE().
getKlassifizierungLEKurz()) &&
fachgebietcode.equals(beRw.getBeRwKlassifizierungFachgebiet().
getBeRwFachgebiet().getFachgebietcode()) &&
beRw.istGueltig()) {
```

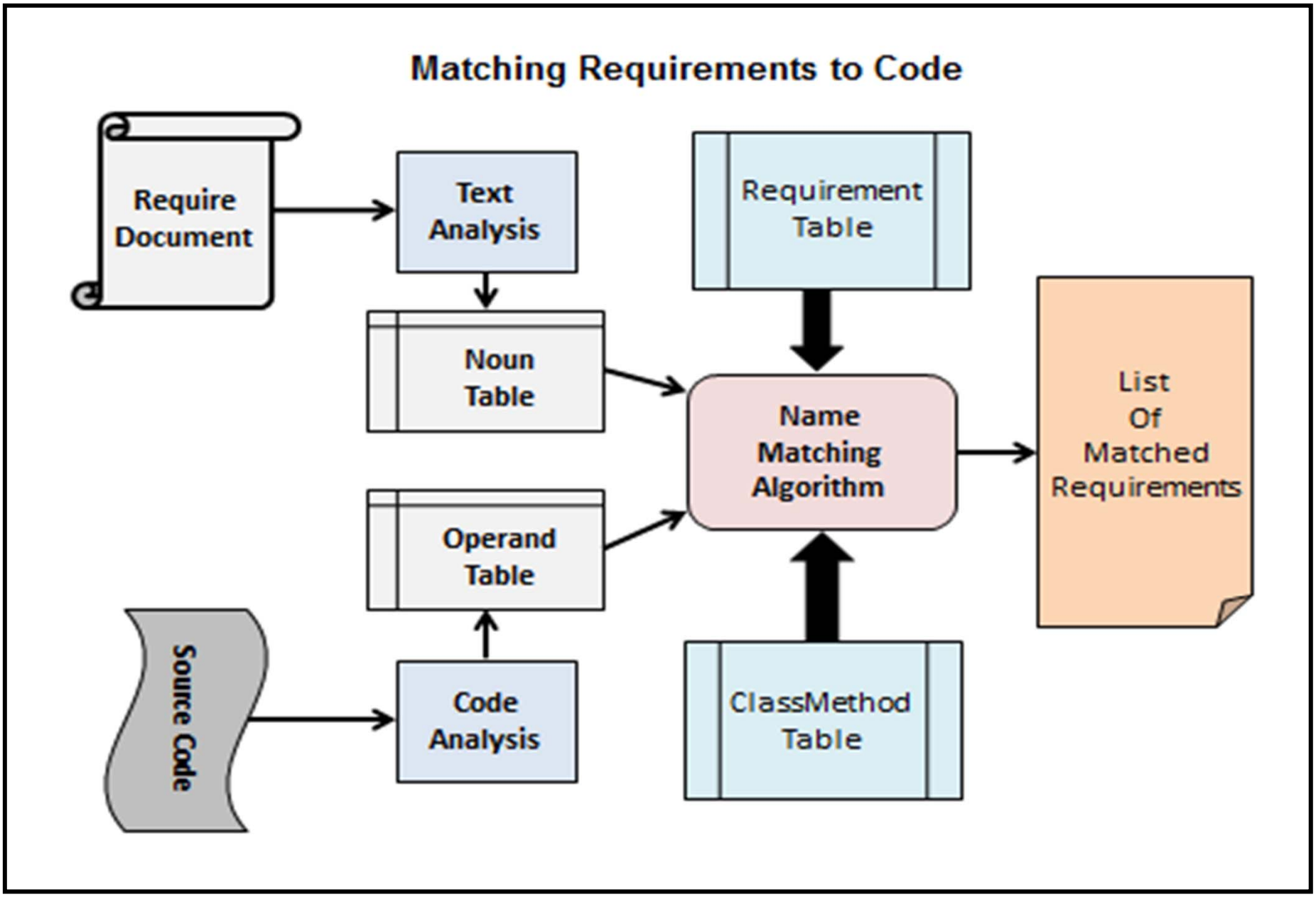
Semantische Kopplung



False Positives = Methoden passen zu vielen Anwendungsfällen

False Negatives = Methoden verwenden andere Datennamen

Zuordnung Codeblöcke zu Anforderungen



Anforderungsabdeckungsbericht


```
+-----------------------------------------------------------+
| S Y S T E M   C O N S I S T E N C Y  M E T R I C   R E P O R T |
| LANGUAGE: GERMAN/JAVA                          DATE: 22.07.15 |
|   SYSTEM: ZVP4                                 PAGE:  2 of  3 |
+-----------------------------------------------------------+
|                E N T I T Y      C O U N T S                 |
+-----------------------------------------------------------+
|   Anzahl spezifizierter Anforderungen        =======>    230   |
|   Anzahl spezifizierter Anwendungsfälle      =======>    266   |
|   Anzahl implementierter Java Methoden       =======>  45262   |
|   Anzahl Anforderung zu Anwendungsfall Links =======>    228   |
|   Anzahl Anwendungsfall zu Java Code Links   =======>    165   |
|   Anzahl nicht zuordnungsbarer Anforderungen =======>      1   |
|   Anzahl nicht zuordnungsbarer Anwendungsfälle =====>    101   |
+-----------------------------------------------------------+
|            C O V E R A G E      M  E T R I C S            |
+-----------------------------------------------------------+
|   Anforderung/Anwendungsfallabdeckungsgrad =======>    0.991   |
|   Anwendungsfall/Codeabdeckungsgrad        =======>    0.620   |
|   Anforderungsabdeckungsgrad               =======>    0.614   |
+-----------------------------------------------------------+
```

Zusammenfassung

- Software Engineering = Ordnung = Disziplin
- Es muss mehr Disziplin in Softwareprojekten herrschen, das beginnt mit der Namensvergebung.
- Voraussetzung für die Zuordnung vom Code zu Anforderungen ist die Verwendung gleicher Datennamen
- Dies setzt wiederum eine strikte Namenskonvention voraus, bzw. eine gemeinsame Ontologie
- Das Ziel ist **Traceability** zwischen allen Entwicklungsstufen der Software
- Nur dann kannst Du ohne großen Testaufwand feststellen wo ein Projekt steht und was ist noch zu tun.