

# Software Engineering

## Requirements Engineering

Prof. Dr. Peter Jüttner

Hochschule Deggendorf

## Inhalt

### 5. Methoden

#### 5.1. Requirements Engineering

##### 5.1.1 Funktionale / Nicht-Funktionale Requirements

##### 5.1.2 Stakeholder

##### 5.1.3 Methodik des Requirements Engineering (incl. Exkurs über Use Cases)

##### 5.1.4 Verwalten von Requirements

##### 5.1.5 Nachverfolgbarkeit (Traceability)

##### 5.1.6 Tools

## Methodik – Requirements Engineering

### Zur Erinnerung

**Anforderungsanalyse** (*Requirements Engineering*) heißt der Prozess des

- Herausfindens (Elicitation),
- Analysierens (Analysis),
- Dokumentierens (Specification) und
- Überprüfens (Validation & Verification)

dieser Anforderungen

Anforderungen beschreiben das **WAS** und nicht das **WIE!**

## Methodik – Requirements Engineering

Wie werden Anforderungen (Requirements) beschrieben?

Natürliche Sprache:

Req. 4.1.2 Strommessung

Für die vom Batteriesensor an den Batteriemonitor übergebenen Stromwerte werden drei verschiedene Messbereiche festgelegt, die den Größenordnungen der Batterieströme bei Fahrzeug-Stillstand (Messbereich 1), im Fahrtzustand (Messbereich 2) und bei Motorstart (Messbereich 3) entsprechen.



## Methodik – Requirements Engineering

Wie werden Anforderungen (Requirements) beschrieben?

Natürliche Sprache:

Req. 5.1.7 Programmiersprache

Durch den Auftraggeber wird die Verwendung der Programmiersprache ANSI C gefordert.

Wenn die Programmiersprache ANSI-C nicht eingesetzt werden kann, muss dies durch den Auftragnehmer begründet und durch den Auftraggeber genehmigt werden.



## Methodik – Requirements Engineering

### In allen Phasen des SW Engineering

- **Lastenheft (Customer Specification)**
  - beschreibt das Produkt aus Kundensicht **ohne** die Lösung vorweg zu nehmen
  - Basis für Angebotserstellung und Auftragsvergabe
- **Pflichtenheft (Product Specification, Software Specification)**
  - beschreibt das Produkt aus Sicht der Entwicklung **ohne** die Lösung vorweg zu nehmen
  - verfeinert das Lastenheft und fügt zusätzliche z.B: entwicklungsinterne Requirements hinzu
- weitere Requirements an SW Bausteine (z.B. Komponenten, Funktionen, Klassen) in SW Architektur und Feindesign

## Inhalt

### 5. Methoden

#### 5.1. Requirements Engineering

##### 5.1.1 Funktionale / Nicht-Funktionale Requirements

##### 5.1.2 Stakeholder

##### 5.1.3 Methodik des Requirements Engineering (incl. Exkurs über Use Cases)

##### 5.1.4 Verwalten von Requirements

##### 5.1.5 Nachverfolgbarkeit (Traceability)

##### 5.1.6 Tools

## Methodik – Requirements Engineering

### Wesentliche Arten von Requirements

#### **Funktionale Requirements**

Requirements, die das Verhalten der SW nach außen beschreiben

#### **Nicht-Funktionale Requirements**

Requirements, die nicht die Außenwirkung der Software betreffen



## Methodik – Requirements Engineering

Funktionale Requirements, z.B.

- **Ein-/Ausgabeverhalten der Software,**
  - „wenn der Schalter betätigt wird, muss das Licht angehen“
  - „es muss die Fakultät der eingelesenen Zahl berechnet werden“
  - „bei Auswahl des Menüs sollen folgende Untermenüs angezeigt werden: ...“
  - „der vom Sensor gelieferte Spannungswert muss über die Leitung x eingelesen werden“

## Methodik – Requirements Engineering

### Nicht-Funktionale Requirements, z.B.

- **Anforderungen an die Entwicklungsumgebung**
  - „das Produkt muss in der Sprache C++ mit dem Compiler XY entwickelt werden“
- **Anforderungen an die Zielumgebung**
  - „das Produkt muss auf den Plattformen Windows XP, Windows Vista und Linux lauffähig sein“
- **Anforderungen an organisatorische Randbedingungen, z.B.**
  - „das Produkt muss gemäß dem V-Modell entwickelt werden“
  - Termin
  - Budget

## Methodik – Requirements Engineering

(Nicht)-Funktionale Requirements (?), z.B.

- **Anforderungen an Performance / Ressourcenverbrauch**
  - „das System muss innerhalb von 10ms reagieren“
  - „die Funktion darf nicht mehr als 500 Byte RAM belegen“
- **Anforderungen an funktionale Sicherheit**
  - „das Produkt muss dem Safety Integrity Level 3 genügen“

## Methodik – Requirements Engineering

- ➔ Achtung: Nicht-funktionale Requirements haben oft einen wesentlichen Einfluss auf Entwurfsentscheidungen!
- ➔ Bemerkung: Die Begriffe „Funktionale Anforderungen“ und „Nicht-funktionale Anforderungen“ sind
  - ➔ nicht eindeutig und daher
  - ➔ „unglücklich“
- ➔ Achtung: Nicht alle Requirements werden explizit dokumentiert, da als selbstverständlich vorausgesetzt!



## Übung

### Nicht explizit dokumentierte Requirements

Überlegen Sie Beispiele von Requirements die nicht (immer) explizit dokumentiert werden!

Begründen Sie Ihr Ergebnis!

10 min, arbeiten Sie ggf. zusammen mit einem Partner



## Inhalt

### 5. Methoden

#### 5.1. Requirements Engineering

##### 5.1.1 Funktionale / Nicht-Funktionale Requirements

##### 5.1.2 Stakeholder

##### 5.1.3 Methodik des Requirements Engineering (incl. Exkurs über Use Cases)

##### 5.1.4 Verwalten von Requirements

##### 5.1.5 Nachverfolgbarkeit (Traceability)

##### 5.1.6 Tools

## Methodik – Requirements Engineering

### Definition „Stakeholder“ (= „Interessenvertreter“)

Alle Personen oder Gruppen, die ein Interesse an der zu entwickelnden Software haben, z.B.:

- Kunde / Auftraggeber
- Nutzer
- Management
- Vertrieb /Marketing
- Entwickler
- Fertigung (bei eingebetteten Systemen)
- Qualitätssicherung
- Externe Institutionen (z.B. Autoversicherungen bei Diebstahlsicherheit)
- Gesetzgeber (z.B. Sicherheitskritische Software)

## Methodik – Requirements Engineering

### Definition „Stakeholder“ (= „Interessenvertreter“)

- ➔ Achtung: Die Interessen verschiedener Stakeholder werden nicht immer identisch sein
- ➔ Dies kann durchaus zu widersprüchlichen Requirements führen, z.B.
  - Der Kunde möchte möglichst wenig für das Produkt bezahlen, der Vertrieb möchte möglichst viel einnehmen.
  - Der Kunde möchte das Produkt möglichst schnell. Die Qualitätssicherung möchte das Produkt in möglichst hoher Qualität



## Methodik – Requirements Engineering

Zur Erfassung aller relevanten Requirements müssen alle möglichen Stakeholder identifiziert (und dokumentiert) werden

Stakeholder können Interesse in verschiedenen Phasen der Entwicklung haben.

Zum Schluss dieses Abschnitts ...

***Noch Fragen ??***

## Inhalt

### 5. Methoden

#### 5.1. Requirements Engineering

##### 5.1.1 Funktionale / Nicht-Funktionale Requirements

##### 5.1.2 Stakeholder

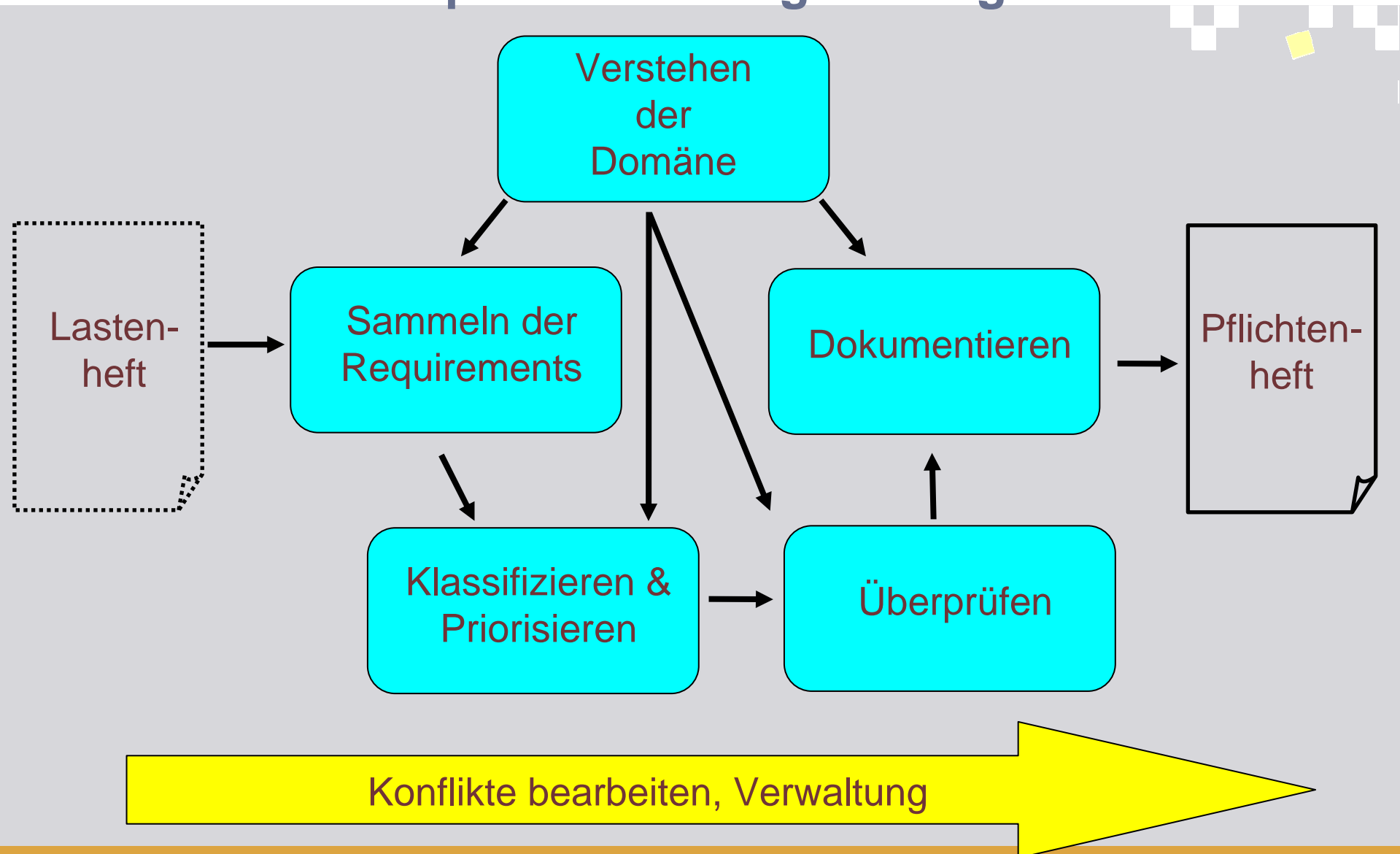
##### 5.1.3 Methodik des Requirements Engineering (incl. Exkurs über Use Cases)

##### 5.1.4 Verwalten von Requirements

##### 5.1.5 Nachverfolgbarkeit (Traceability)

##### 5.1.6 Tools

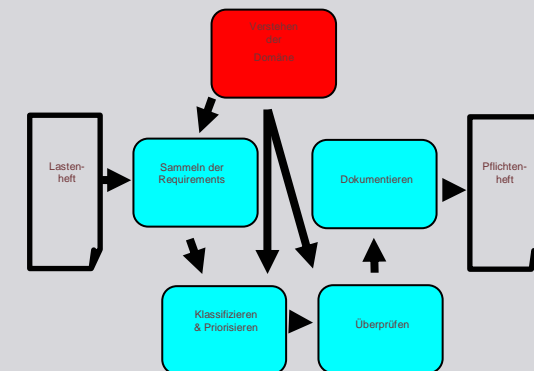
## Methodik – Requirements Engineering



## Methodik – Requirements Engineering

### Verstehen der Domäne

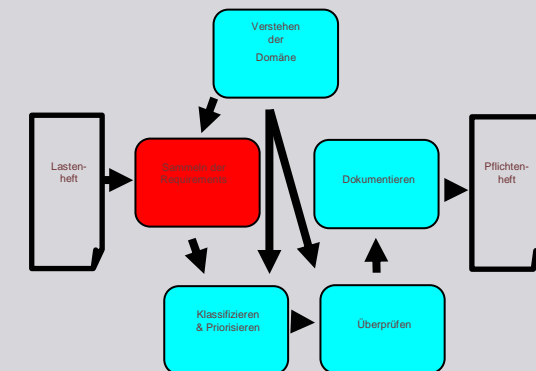
- Domäne = Einsatz- und Anwendungsbereich der zu entwickelnden Software (z.B. Automotive, Medizin, Luftfahrt, Büro, Telekommunikation)
- erleichtert bzw. ermöglicht das Verstehen der Anforderungen aus dem Lastenheft
- erleichtert die Kommunikation mit dem Auftraggeber
- erleichtert das weitere Bearbeiten der Anforderungen und das Umsetzen in ein Pflichtenheft
- erleichtert das Erstellen eines Angebots an den Auftraggeber



## Methodik – Requirements Engineering

### Sammlung der Requirements (Elicitation)

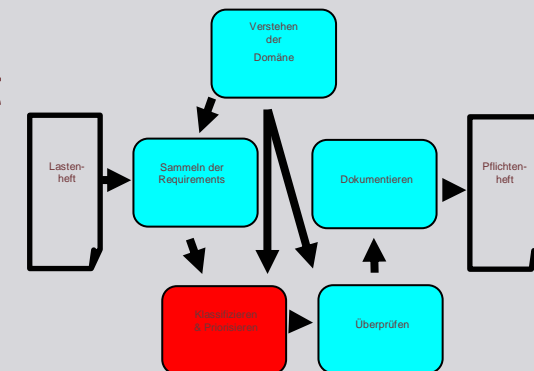
- Übernahme der Anforderungen aus dem Lastenheft (sofern vorhanden)
- Brainstorming (mit allen Stakeholdern)
- Interviewtechnik (mit allen Stakeholdern)
- Ermittlung von Anwendungsfällen (Use Cases)
- Übernahme der Anforderungen aus einem Vorgängerprojekt
- Auswerten von Rückmeldungen der Anwender eines existierenden Produkts
- Auswerten von Testergebnissen
- Marktanalyse (Konkurrenzprodukte)



## Methodik – Requirements Engineering

### Klassifizieren der Requirements (Analysis)

- Klassifizierung bzgl. Abhängigkeit zwischen Requirements, z.B.
  - welche bedingen sich?
  - welche schließen sich gegenseitig aus?
- Klassifizierung bzgl. Zusammengehörigkeit von Requirements, z.B.
  - welche gehören zusammen?
  - welche lassen sich nur gemeinsam realisieren?
- Rollenbezogene Klassifizierung
  - zu welcher Rolle gehört welches Requirement



## Methodik – Requirements Engineering

### Beispiel Rollenbezogene Klassifizierung (Restaurant - Bierausschank)

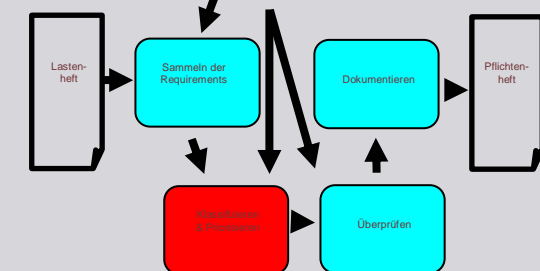
Requirements des Endkunden (Gast), z.B.  
„Das Bier muss richtig temperiert sein“

Requirements des Lieferanten, z.B.  
„Der Zugang zum Restaurant muss breit genug für Bierfässer sein“



Requirements des Wirts, z.B.  
„Bier muss in ausreichender Menge vorhanden sein“

Requirements des Gesetzgebers, z.B.  
„Ausschank von Bier an Personen unter 16 Jahren ist nicht gestattet“





## Methodik – Requirements Engineering

### Beispiel Rollenbezogene Klassifizierung (Automobilelektronik - Lichtsteuerung)

Requirements des Endkunden  
(Autofahrer), z.B.  
„Bei Betätigung des Lichtschalters  
muss das Fahrlicht angehen“

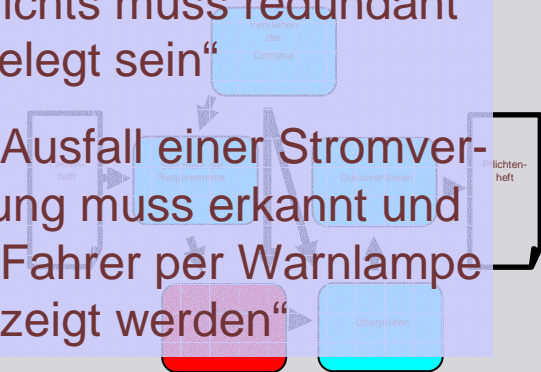
Requirements des Lieferanten, z.B.  
„die Software der Lichtsteuerung  
muss in der Fertigung flashbar sein“

Requirements des  
Automobilherstellers, z.B.  
„der Fehlerspeicher der  
Lichtsteuerung muss in der  
Werkstatt auslesbar sein“



Requirements des Gesetz-  
gebers, z.B.  
„die Stromversorgung des  
Fahrlichts muss redundant  
ausgelegt sein“

„Der Ausfall einer Stromver-  
sorgung muss erkannt und  
dem Fahrer per Warnlampe  
angezeigt werden“



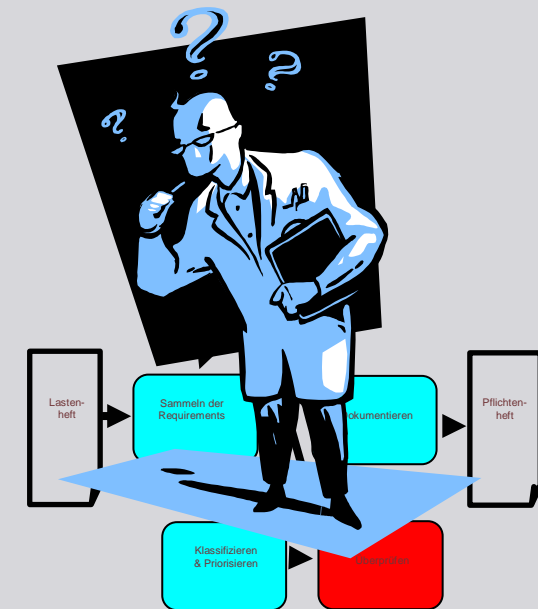


## Methodik – Requirements Engineering

### Überprüfen (Validation & Verification)

Sind die richtigen Requirements spezifiziert ?

Sind die Requirements richtig spezifiziert ?

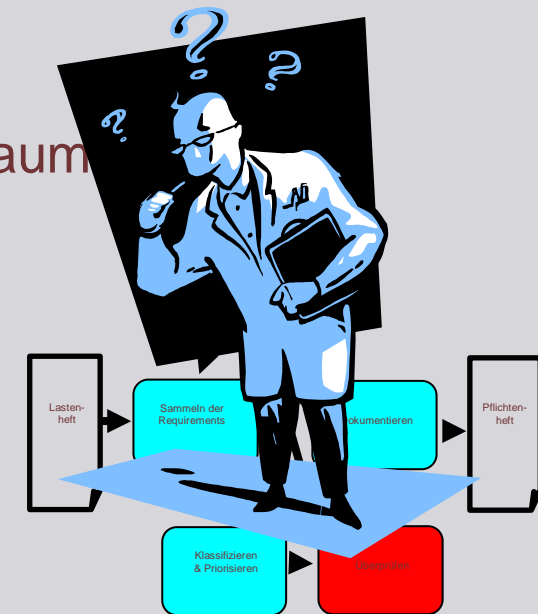


## Methodik – Requirements Engineering

### Überprüfen (Validation & Verification)

Sind die Requirements

- Adäquat, d.h. der Aufgaben- und Problemstellung angemessen?
- Vollständig, d.h. es fehlen keine Anforderungen?
- überdefiniert, d.h. gibt es überflüssige Anforderungen
- Verständlich
- Eindeutig, d.h. es gibt keinen Interpretationsspielraum
- Konsistent d.h. widerspruchsfrei?
- Testbar, d.h. kann für jede Anforderung ein entsprechender Testfall gefunden werden?
- Prüfbar?

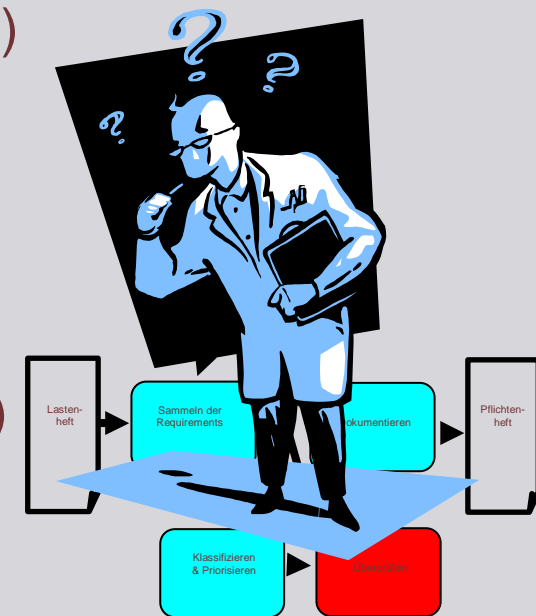


# Methodik – Requirements Engineering

## Überprüfen (Validation & Verification)

### Methoden

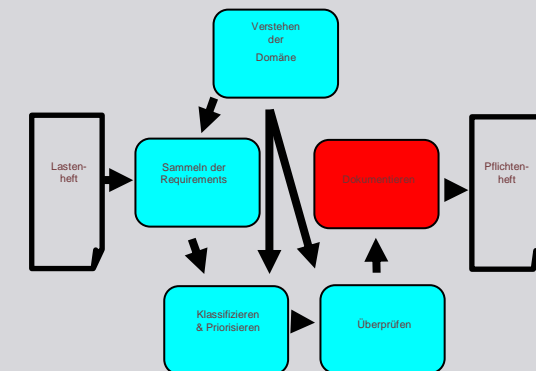
- Reviews -> Kap. 5.5
- Simulation
  - formal beschriebene Requirements mittels entsprechender Tools (z.B. Statemate, Matlab)
  - „gedanklich“
- Prototyping
- Testen
  - lässt sich zu jedem Requirement (mindestens) ein Testfall definieren?



## Methodik – Requirements Engineering

### Dokumentation - Inhalt der Beschreibung:

- Funktion, z.B. wie werden Eingabedaten bearbeitet, wie ist die Struktur der Daten
- Leistung, z.B. Datendurchsatz
- Kommunikation, z.B. Bedienoberfläche, Bussysteme, Schnittstellen
- Fehlerfälle, z.B. wie reagiert die SW auf fehlerhafte bzw. unplausible Eingaben oder fehlerhafte Umgebungen
- Qualitätsaspekte
- Sonstige Randbedingungen



## Methodik – Requirements Engineering

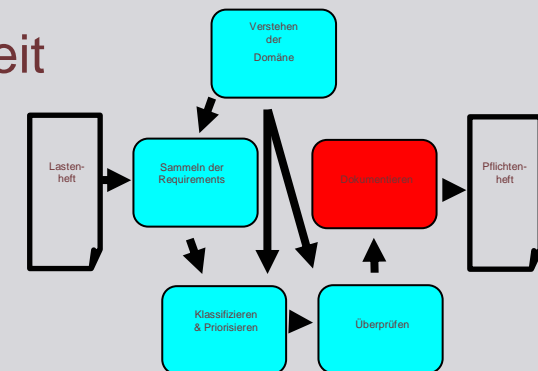
### Beschreibung durch natürliche Sprache:

#### Vorteile:

- verständlich für Auftraggeber und Auftragnehmer
- eignet sich gut, einen Überblick über die zu entwickelnde Software zu geben
- ausdrucksstark

#### Nachteil:

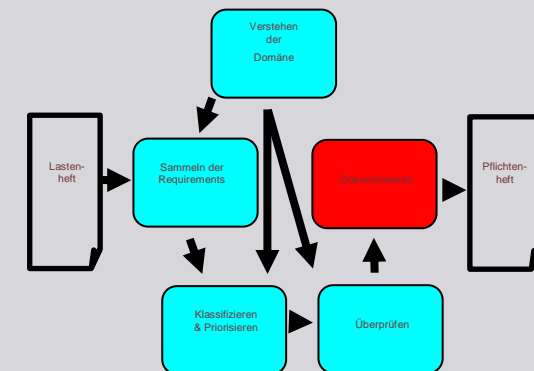
- erlaubt Interpretationen (mehrdeutig)
- nicht maschinell auf Konsistenz oder Vollständigkeit überprüfbar
- unübersichtlich, falls als alleiniges Beschreibungsmittel verwendet



## Methodik – Requirements Engineering

### Beschreibung durch natürliche Sprache:

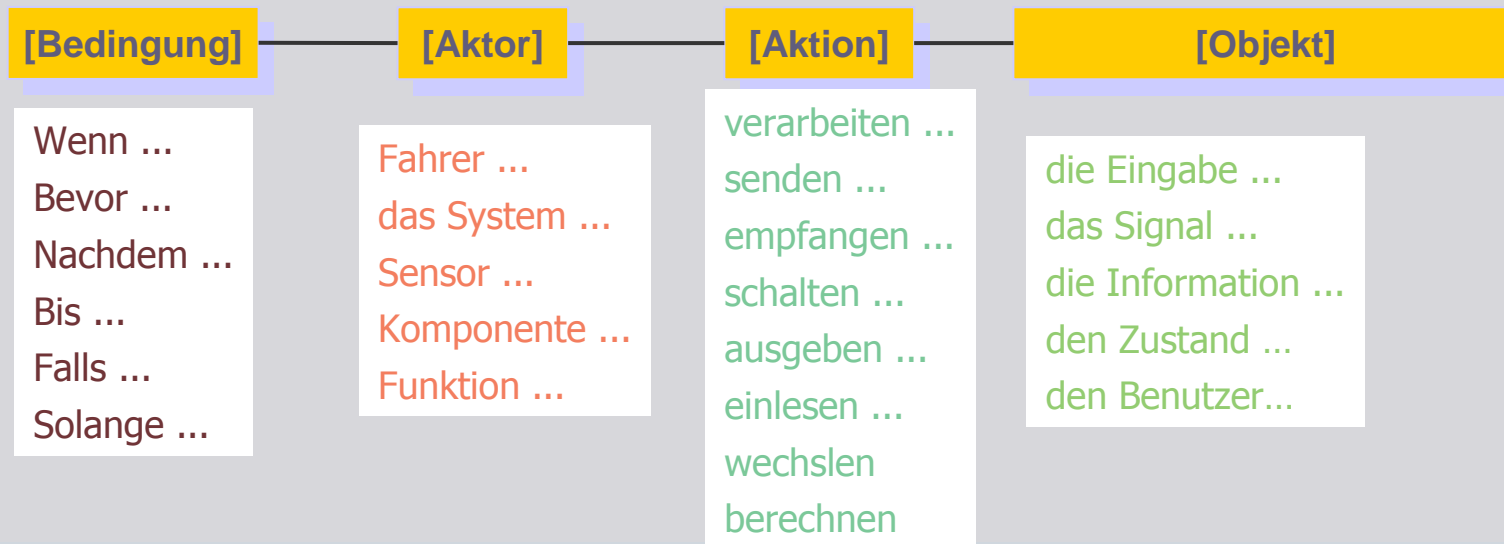
- ➔ Sollte (bzw. muss) immer verwendet werden in geeigneter Weise (Reduktion von Mehrdeutigkeiten)
  - ➔ geeignete Strukturierung
  - ➔ geeignete Formulierungen
  - ➔ Glossar (Lexikon der verwendeten Begriffe)
  - ➔ Abkürzungsverzeichnis
- ➔ Ergänzen durch weitere Beschreibungsmethoden, z.B.
  - ➔ Diagramme, Tabellen, Bilder
  - ➔ formale Beschreibungen





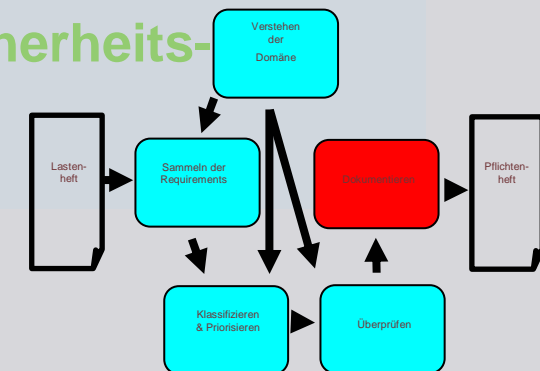
## Methodik – Requirements Engineering

### Wie werden gute textuelle Requirements beschrieben?



#### Beispiel Reifendruckkontrolle:

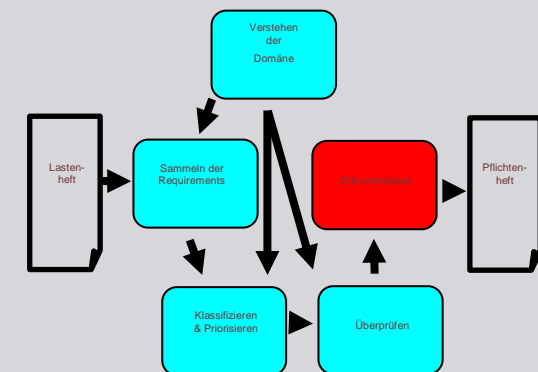
Während der Fahrt **muss das System** einen sicherheitsrelevanten Druckverlust in Reifen durch Einschalten der Warnlampe anzeigen



## Methodik – Requirements Engineering

### Wie werden gute textuelle Requirements beschrieben?

- **Wenig (oder kein) Interpretationsspielraum**
  - „das System muss möglichst schnell reagieren“ **oder**
  - „das System muss in höchstens 10ms wie beschrieben reagieren“
  - „das System muss möglichst alle Daten zu den Versicherten ausdrucken“ **oder**
  - „das System muss die Daten Name, Vorname, Geburtsdatum und Adresse zu den Versicherten ausdrucken“



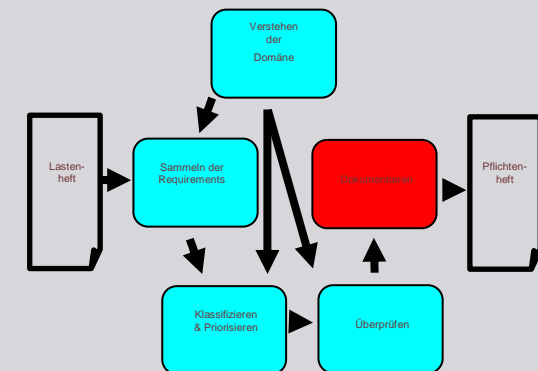
## Methodik – Requirements Engineering

### Wie werden gute textuelle Requirements beschrieben?

#### Atomar und eindeutig identifizierbar

- „die Alarmanlage muss im Fall eines Einbruchs das Blinklicht und die Sirene einschalten und über Telefonleitung die Polizei benachrichtigen und den Wachmann über Handy alarmieren“

oder

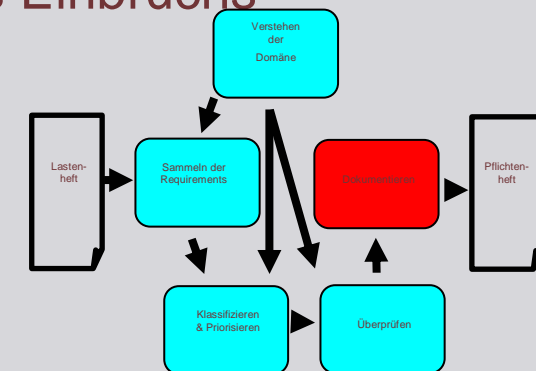


## Methodik – Requirements Engineering

Wie werden gute textuelle Requirements beschrieben?

### Atomar und eindeutig identifizierbar

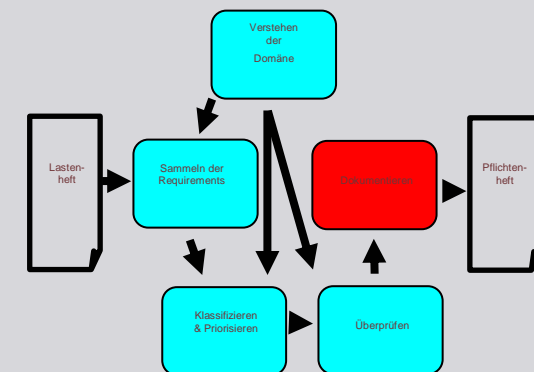
- *Req. Alarm 1:* die Alarmanlage muss im Fall eines Einbruchs das Blinklicht einschalten
- *Req. Alarm 2:* die Alarmanlage muss im Fall eines Einbruchs die Sirene einschalten
- *Req. Alarm 3:* die Alarmanlage muss im Fall eines Einbruchs Telefonleitung die Polizei benachrichtigen
- *Req. Alarm 4:* die Alarmanlage muss im Fall eines Einbruchs den Wachmann über Handy alarmieren



## Methodik – Requirements Engineering

### Wie werden gute textuelle Requirements beschrieben?

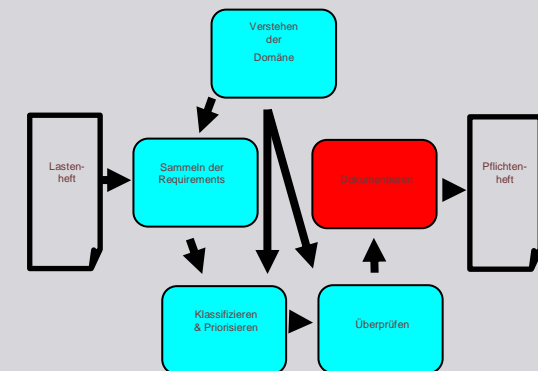
- Eindeutig identifizierbar
  - ➔ Vermeidung von Mehrdeutigkeiten
  - ➔ Verbesserte Nachverfolgbarkeit (Traceability)
  - ➔ aber: Sicherstellen, dass der Gesamtzusammenhang verständlich bleibt, z.B. durch Einleitungskapitel, Kommentare



## Methodik – Requirements Engineering

### Wie werden gute textuelle Requirements beschrieben?

- Präzise beschreiben
  - ➔ Substantive anstatt „man“, „es“
  - ➔ Definierte Bedeutung von „muss“, „soll“, „kann“, „darf“
  - ➔ Quantifizierte Aussagen
  - ➔ Angemessene Detaillierung (je mehr Details, desto mehr Aufwand bei der Beschreibung)



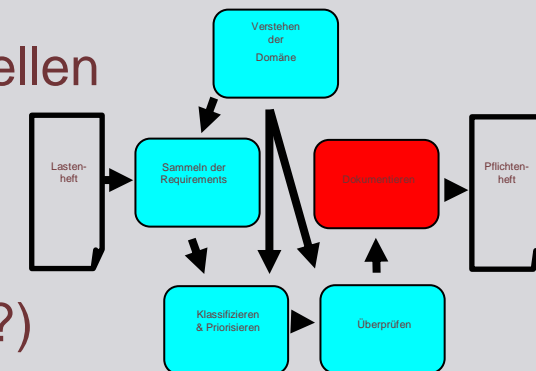
## Methodik – Requirements Engineering

### Beispiel Software Requirements Specification angelehnt an IEEE 830 (Quelle Wikipedia)

- Name des Softwareprodukts
- Name des Herstellers
- Versionsdatum des Dokuments und / oder der Software

#### 1. Einleitung

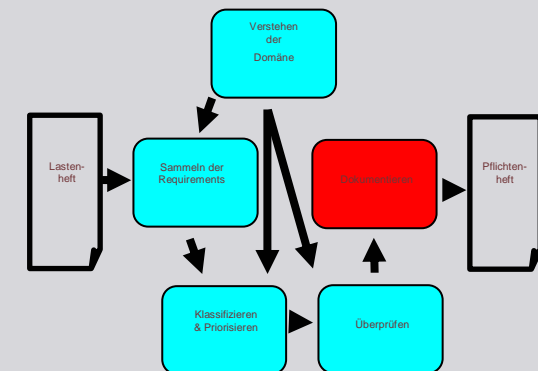
1. Zweck (des Dokuments)
2. Ziel (des Softwareprodukts)
3. Verweise auf sonstige Ressourcen oder Quellen
4. Erläuterungen zu Begriffen und / oder Abkürzungen
5. Übersicht (Wie ist das Dokument aufgebaut?)



## Methodik – Requirements Engineering

### Beispiel Software Requirements Specification angelehnt an IEEE 830 (Quelle Wikipedia)

2. Allgemeine Beschreibung (des Softwareprodukts)
  1. Produkt Perspektive (zu anderen Softwareprodukten)
  2. Produktfunktionen (eine Zusammenfassung und Übersicht)
  3. Benutzermerkmale (Informationen zu erwarteten Nutzern, z.B. Bildung, Erfahrung, Sachkenntnis)
  4. Einschränkungen (für den Entwickler)
  5. Annahmen und Abhängigkeiten (nicht Realisierbares und auf spätere Versionen verschobene Eigenschaften)

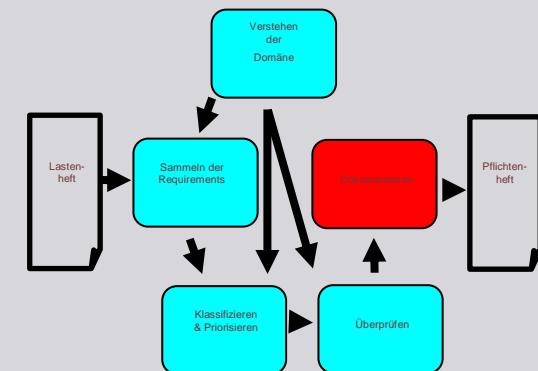




## Methodik – Requirements Engineering

### Beispiel Software Requirements Specification angelehnt an IEEE 830 (Quelle Wikipedia)

3. Spezifizierte Anforderungen (im Gegensatz zu 2.)
  1. funktionale Anforderungen (Stark abhängig von der Art des Softwareprodukts)
  2. nicht-funktionale Anforderungen
  3. externe Schnittstellen
  4. Design Constraints
  5. Anforderungen an Performance
  6. Qualitätsanforderungen
  7. Sonstige Anforderungen



## Übung

### Kneipenrequirements

Sie möchten in Deggendorf eine gut laufende Studentenkneipe eröffnen.

Überlegen Sie sich (max 10) Anforderungen an ihre Kneipe!  
Priorisieren sie anschließend ihre Requirements

10 min, arbeiten Sie ggf. zusammen mit einem Partner



## Methodik – Requirements Engineering

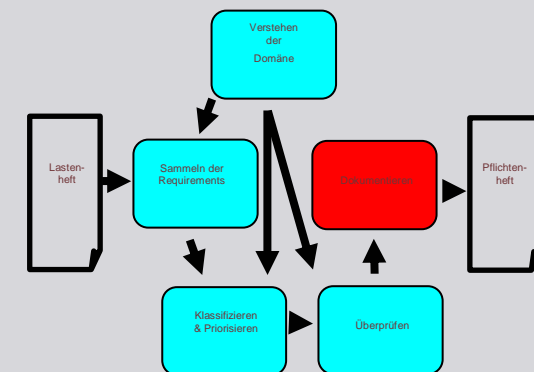
### Modellierung von Requirements

Ergänzung und/oder Ersatz der natürlichsprachlichen Beschreibung

**Ziel:** Anforderungen nicht (nur) als eine Sammlung von Sätzen in natürlicher Sprache dokumentieren, sondern ein anwendungsnahes Modell **der Aufgabenstellung** erstellen (keine Lösung!)

Formale oder teilformale Beschreibungen meist unter Verwendung grafischer beschriebener Modelle

- Hauptsächlich zur Darstellung funktionaler Requirements

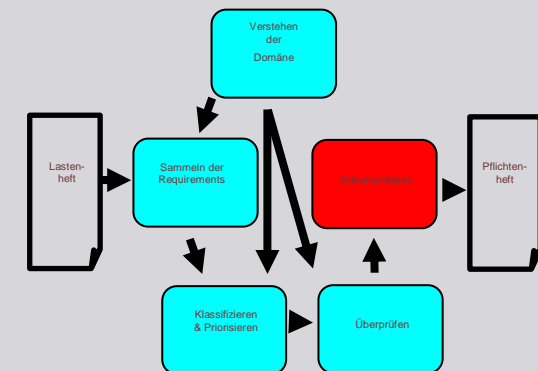




## Methodik – Requirements Engineering

### Modellierung von Requirements

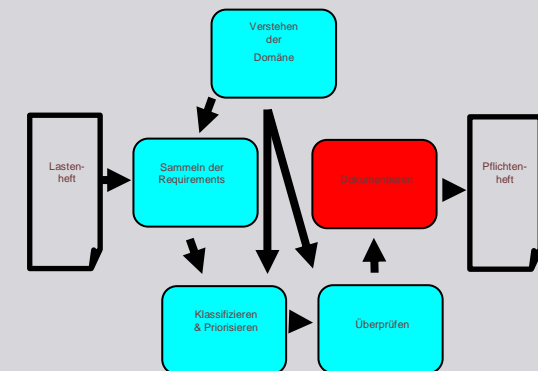
- + problemnahe Beschreibung
- + eindeutig, formal
- + formal überprüfbar
- + teilweise simulierbar
- + „sanfter“ Übergang von Requirements zum Entwurf
- + teilweise Code aus Modellen generierbar



## Methodik – Requirements Engineering

### Modellierung von Requirements

- erfordert detaillierte Kenntnisse der verwendeten Sprache und Tools
- erfordert detaillierte Erfahrung mit Modellierung
- Tools (teilweise) sehr teuer
- weckt zu hohe Erwartungen

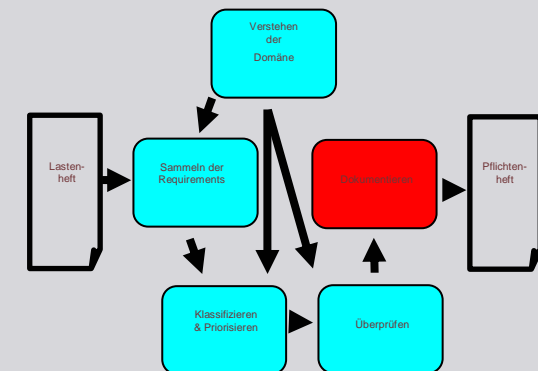




# Methodik – Requirements Engineering

# Exkurs: UML Use Cases

- Bestandteil der UML
- Dient zur Modellierung von Anforderungen (Was soll mein System bzw. meine SW leisten?)
- Beschreibt das System aus Sicht der Nutzer (Akteure) in Anwendungsfällen
- Nutzer = Person, anderes System
- Use Case wird beschrieben durch Use Case Diagramm, ergänzt durch textuelle Beschreibung, Sequenz Diagramme, Zustandsautomaten (s. a. Kap. 5.2)

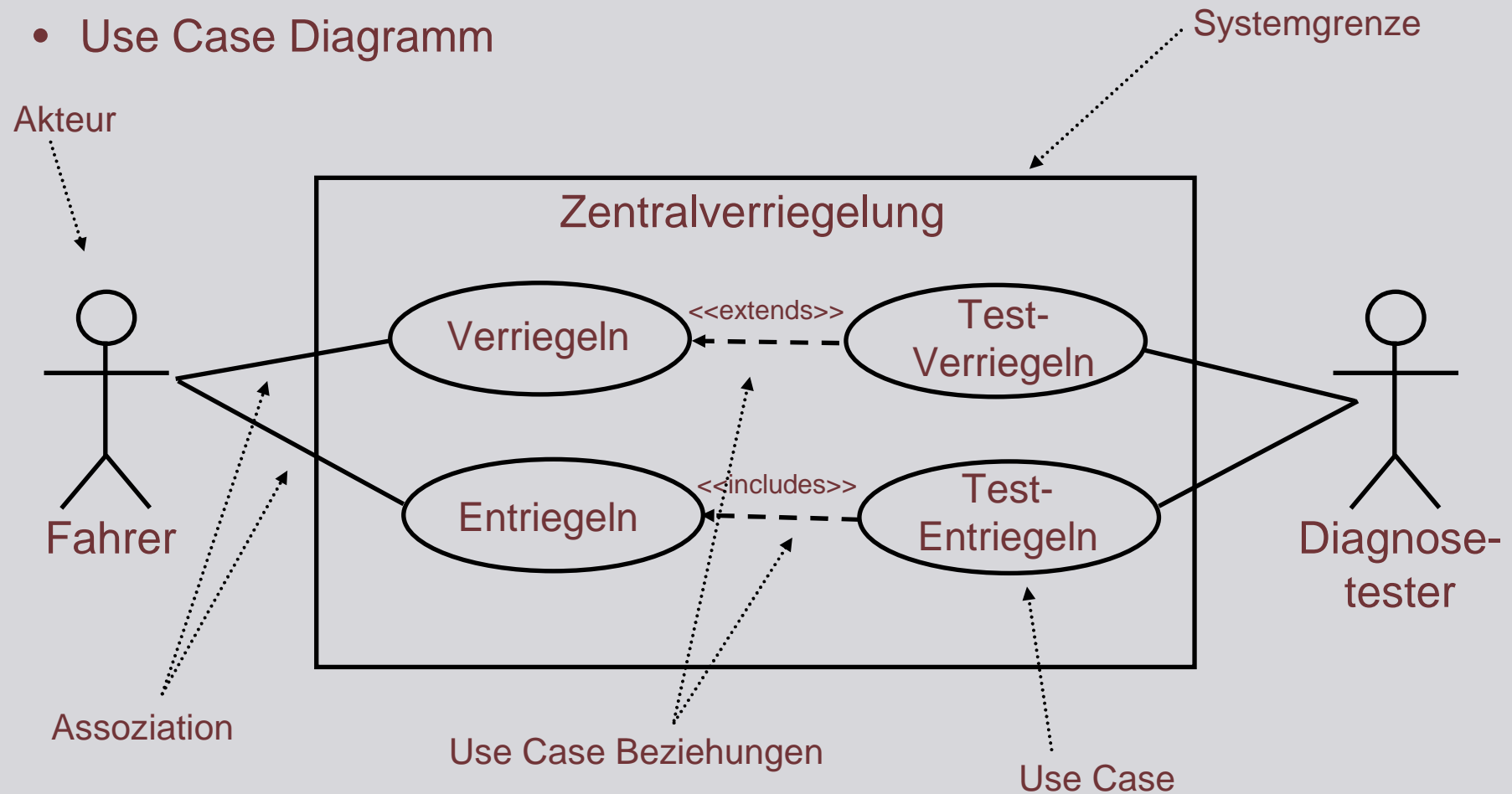




## Methodik – Requirements Engineering

### Exkurs: UML Use Cases

- Use Case Diagramm



Zum Schluss dieses Abschnitts ...

***Noch Fragen ??***

## Inhalt

### 5. Methoden

#### 5.1. Requirements Engineering

##### 5.1.1 Funktionale / Nicht-Funktionale Requirements

##### 5.1.2 Stakeholder

##### 5.1.3 Methodik des Requirements Engineering (incl. Exkurs über Use Cases)

##### 5.1.4 Verwalten von Requirements

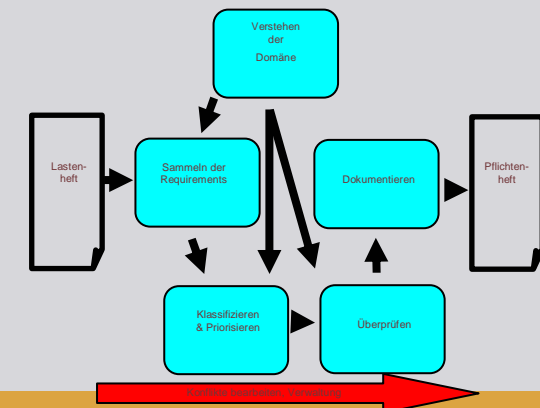
##### 5.1.5 Nachverfolgbarkeit (Traceability)

##### 5.1.6 Tools

## Methodik – Requirements Engineering

### Verwaltung von Requirements (Requirements Management)

- Änderung (modification, change management)
- Verfolgung (traceability)
- Freigabe (baselining, release management)

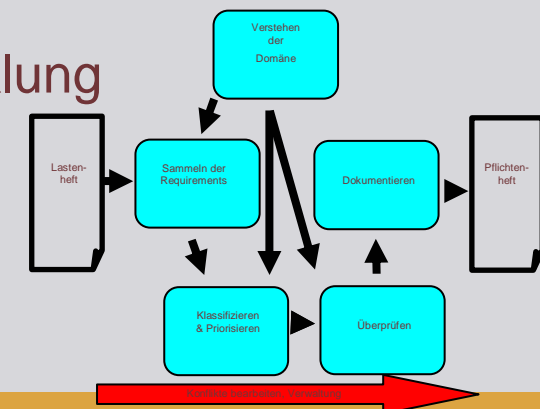


## Methodik – Requirements Engineering

### Änderungen von Requirements verwalten

#### Probleme:

- Requirements unterliegen Änderungen, z.B.
  - durch den Auftraggeber
  - technischer Fortschritt
  - interne Änderungen
- Je später ein Requirement geändert wird, desto teurer ist die Umsetzung
- Requirements sollen für eine „ungestörte“ Entwicklung möglichst stabil sein.
- Änderungen müssen daher kontrolliert zugelassen werden

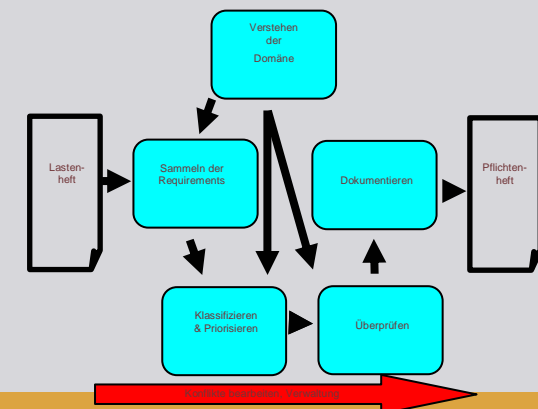


## Methodik – Requirements Engineering

### Änderungen von Requirements verwalten

#### Vorgehensweise:

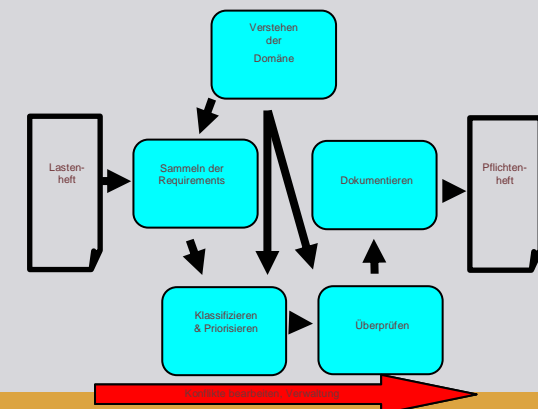
- Definition eines Änderungsprozesses (Change Management) für Requirements
- Versionierung von Requirements
- Methodik s. Kap. 5.7



## Methodik – Requirements Engineering

### Freigabe (baselining, release management)

- Die Menge der in einem Entwicklungszyklus umzusetzenden Requirements festschreiben
- Menge der umgesetzten Requirements in einer Baseline festhalten



Zum Schluss dieses Abschnitts ...

***Noch Fragen ??***



## Inhalt

### 5. Methoden

#### 5.1. Requirements Engineering

##### 5.1.1 Funktionale / Nicht-Funktionale Requirements

##### 5.1.2 Stakeholder

##### 5.1.3 Methodik des Requirements Engineering (incl. Exkurs über Use Cases)

##### 5.1.4 Verwalten von Requirements

##### 5.1.5 Nachverfolgbarkeit (Traceability)

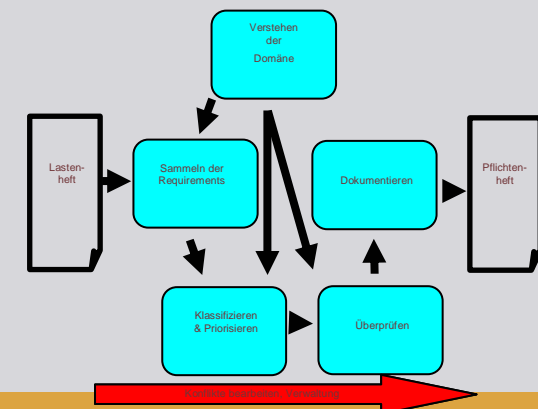
##### 5.1.6 Tools

## Methodik – Requirements Engineering

### Verfolgung von Requirements (Traceability)

#### Ziele:

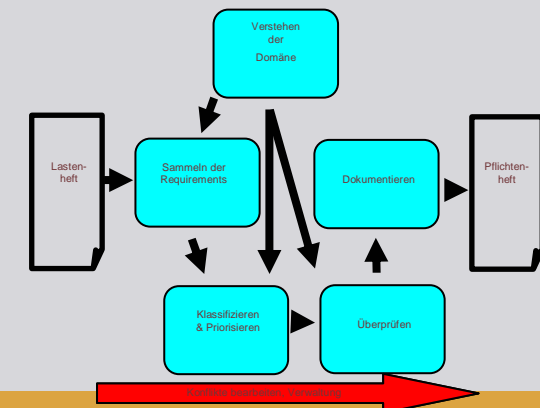
- Sicherstellen, dass alle zu implementierenden Requirements auch implementiert wurden
- Sicherstellen, dass alle umgesetzten Requirements getestet werden
- Auswirkungen von Änderungen in den Requirements verfolgen können



## Methodik – Requirements Engineering

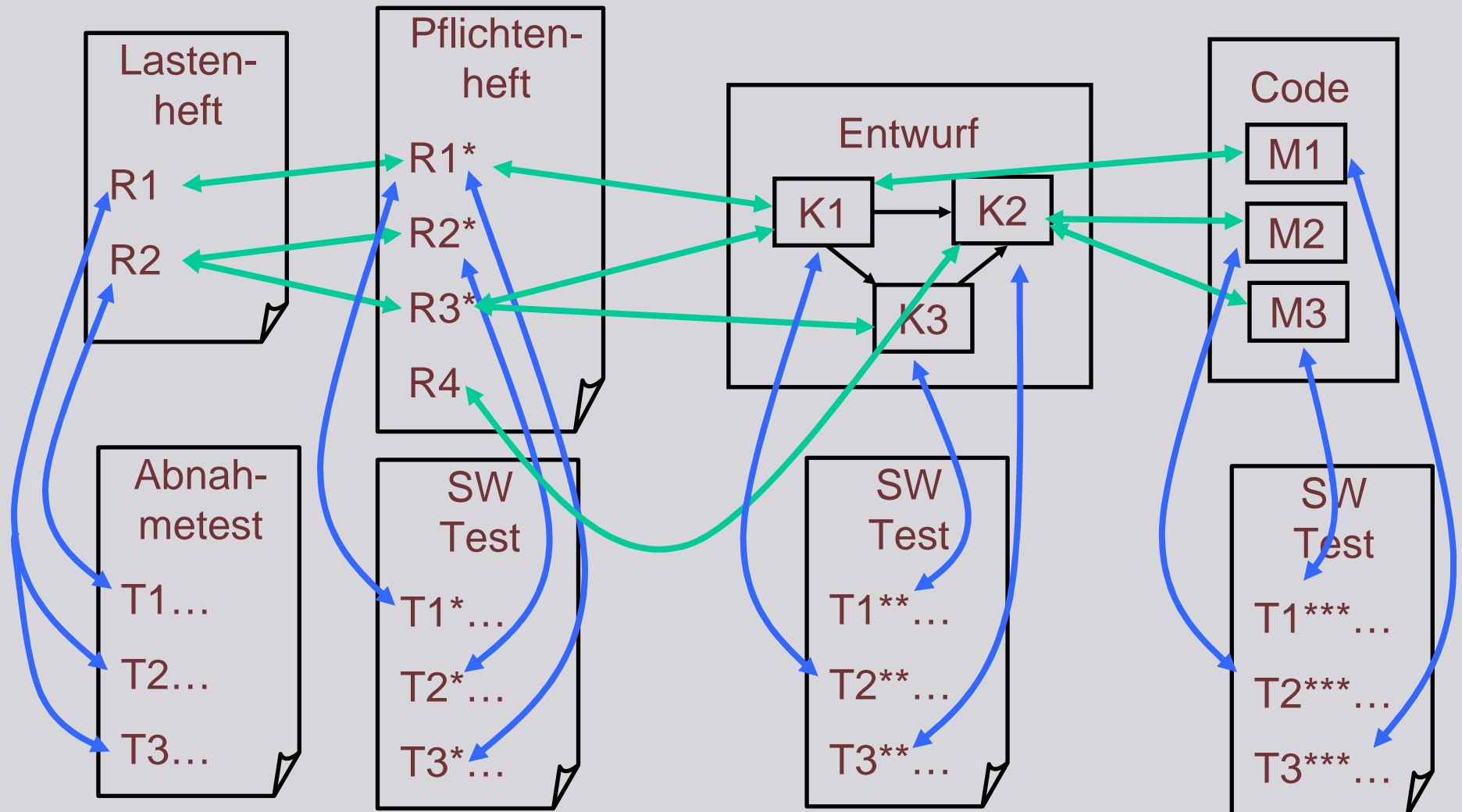
### Verfolgung von Requirements (Traceability)

- Welche Beziehung besteht zwischen Requirements?
- Wo (in der Software) ist welches Requirement umgesetzt?
- Durch welche Testfälle wird welches Requirement getestet?
- Welche Teile der Software sind von der Änderung eines Requirements betroffen?
- Traceability in beiden Richtungen, d.h. von den Requirements zu Implementierung und zu den Testfällen und umgekehrt.



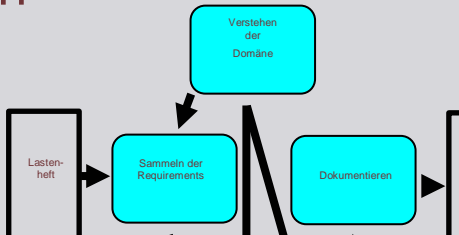
## Methodik – Requirements Engineering

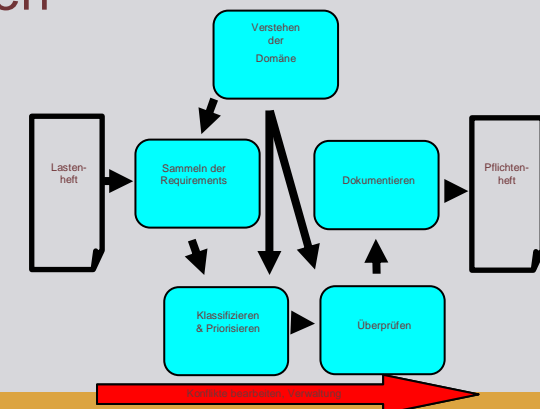
### Verfolgung von Requirements (Traceability)



# Methodik – Requirements Engineering

## Verfolgung von Requirements (Traceability)

- Der Grad der erforderlichen Traceability kann von Projekt zu Projekt variieren
    - Prototypen ggf. ohne Traceability
    - SW, die wieder verwendet werden soll oder SW, die sich häufig ändert, erfordert einen höheren Grad an Traceability
    - Normen (z.B. IEC 61508 für sicherheitskritische Software) können einen sehr hohen Grad erfordern
    - Der Aufwand zur Pflege kann sehr hoch werden
    - „Extreme“ Traceability wie „In welcher Zeile des Codes ist das Requirement X umgesetzt?“ macht keinen Sinn!
- 
- ```
graph LR; A[Lastenheft] --> B[Sammeln der Requirements]; B --> C[Verstehen der Domäne]; C --> D[Dokumentieren]; D --> E[ ]; style B fill:#00FFFF,stroke:#000,stroke-width:2px; style C fill:#00FFFF,stroke:#000,stroke-width:2px; style D fill:#00FFFF,stroke:#000,stroke-width:2px;
```
- Das Diagramm zeigt den Prozess des Requirements Engineering. Es beginnt mit einem grauen Rechteck 'Lastenheft', das auf ein hellblaues Rechteck 'Sammeln der Requirements' führt. Von dort führt ein Pfeil nach oben zu einem hellblauen Rechteck 'Verstehen der Domäne'. Ein weiterer Pfeil führt von 'Verstehen der Domäne' nach unten zu einem hellblauen Rechteck 'Dokumentieren'. Ein abschließender Pfeil führt von 'Dokumentieren' nach rechts zu einem weiteren grauen Rechteck, das teilweise sichtbar ist.

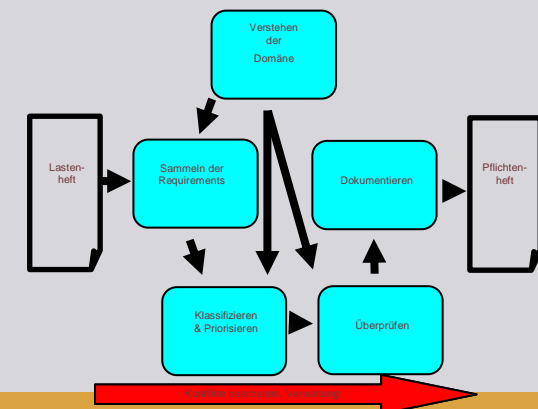


## Methodik – Requirements Engineering

### Verfolgung von Requirements (Traceability)

→ Die Umsetzung der Traceability muss den projektspezifischen Gegebenheiten angepasst werden

- Toolgestützt, z.B. Links in DOORS, evtl. problematisch, falls nicht zusammenpassende Tools verwendet werden
- Traceability Tabellen, in vielen Fällen ausreichend
- Kapitelweise Traceability innerhalb von Dokumenten



Zum Schluss dieses Abschnitts ...

***Noch Fragen ??***

## Inhalt

### 5. Methoden

#### 5.1. Requirements Engineering

##### 5.1.1 Funktionale / Nicht-Funktionale Requirements

##### 5.1.2 Stakeholder

##### 5.1.3 Methodik des Requirements Engineering (incl. Exkurs über Use Cases)

##### 5.1.4 Verwalten von Requirements

##### 5.1.5 Nachverfolgbarkeit (Traceability)

##### 5.1.6 Tools



## Methodik – Requirements Engineering

### Requirements Engineering Tools

- für „einfache“ Anwendungen können die Office-Tools Word oder Excel verwendet werden
- für komplexe Anwendungen reichen diese Tools nicht mehr aus

#### ➔ datebank-basierte Tools

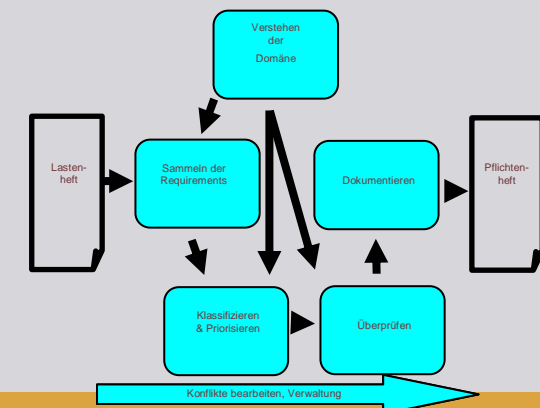
➔ komfortable Sortier- und Suchmöglichkeiten

➔ Verknüpfung von Requirements (Traceability) untereinander und mit anderen Artefakten

➔ Versionierung

➔ Bearbeitungszustand

#### ➔ Wiki-basierte Tools (Trend?)

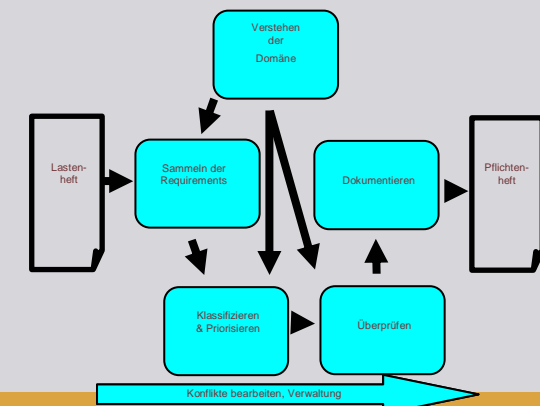


## Methodik – Requirements Engineering

### Requirements Engineering Tools\*)

- Caliber-RM (Borland)
- CARE (Sophist Group)
- DOORS (Telelogic)
- in-Step (microTOOL)
- RequisitePro (IBM / Rational)
- ...

\*) Die Reihenfolge enthält keine Wertung, die Aufzählung erhebt keinen Anspruch auf Vollständigkeit



Zum Schluss dieses Abschnitts ...

***Noch Fragen ??***