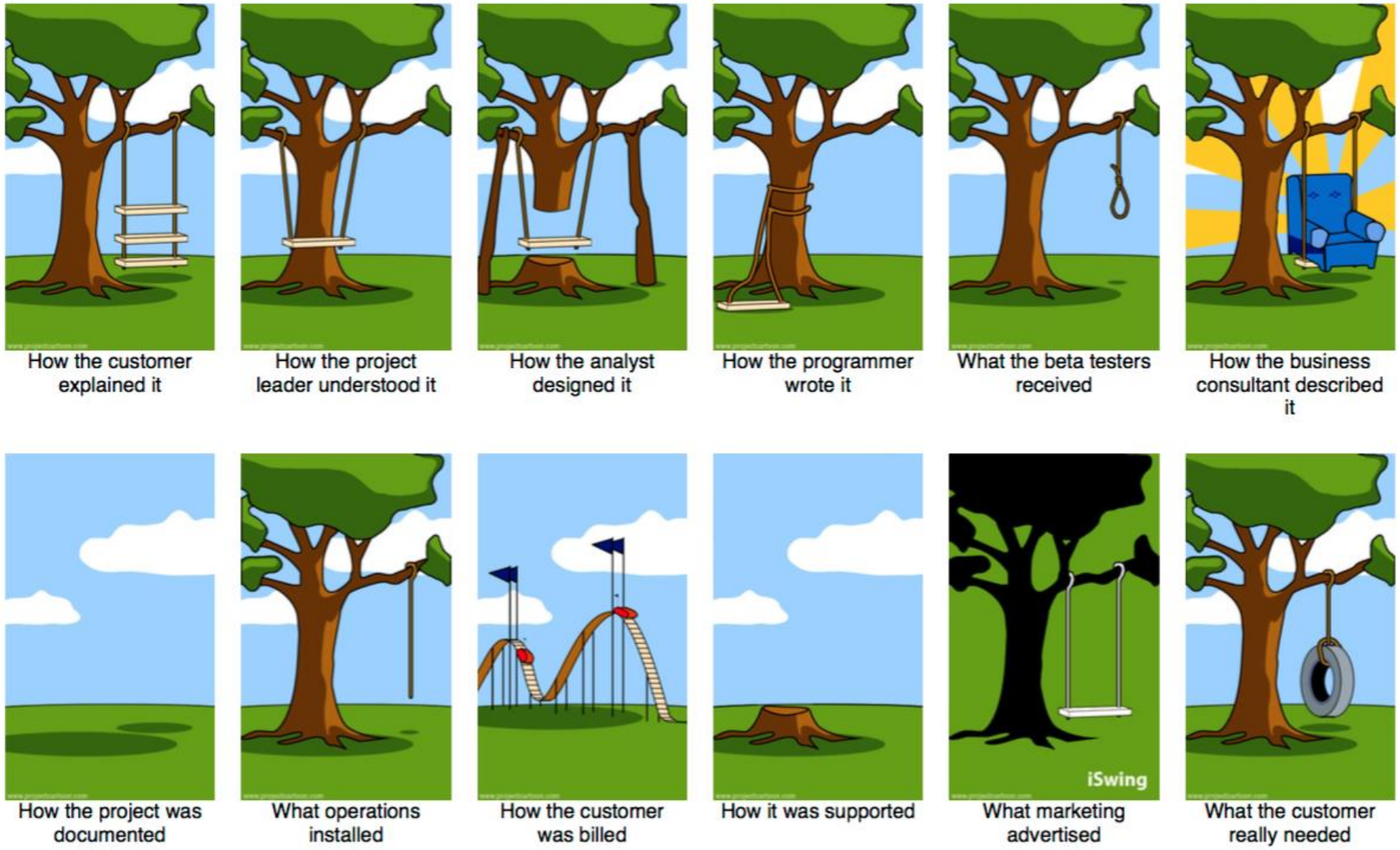
Hochschule Konstanz
Fakultät Informatik

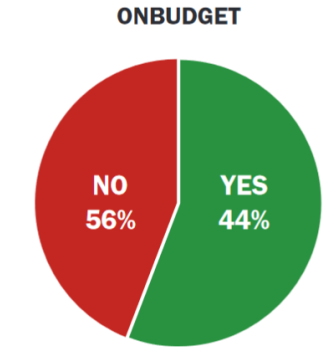
Requirements Engineering

Methodischer Umgang mit Anforderungen an Software-Systeme

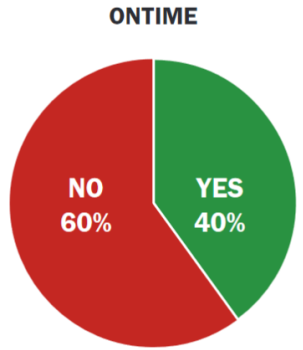


CHAOS-Studie

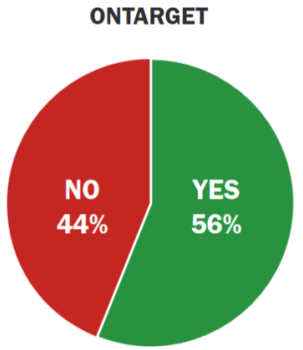
- systematische Bewertung von mehr als 25.000 IT-Projekten



The percentage of projects that were OnBudget from FY2011–2015 within the new CHAOS database.



The percentage of projects that were OnTime from FY2011–2015 within the new CHAOS database.



The percentage of projects that were OnTarget from FY2011–2015 within the new CHAOS database.

TRADITIONAL RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	39%	37%	41%	36%	36%
CHALLENGED	39%	46%	40%	47%	45%
FAILED	22%	17%	19%	17%	19%

https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

Gründe für das Scheitern von IT-Projekten

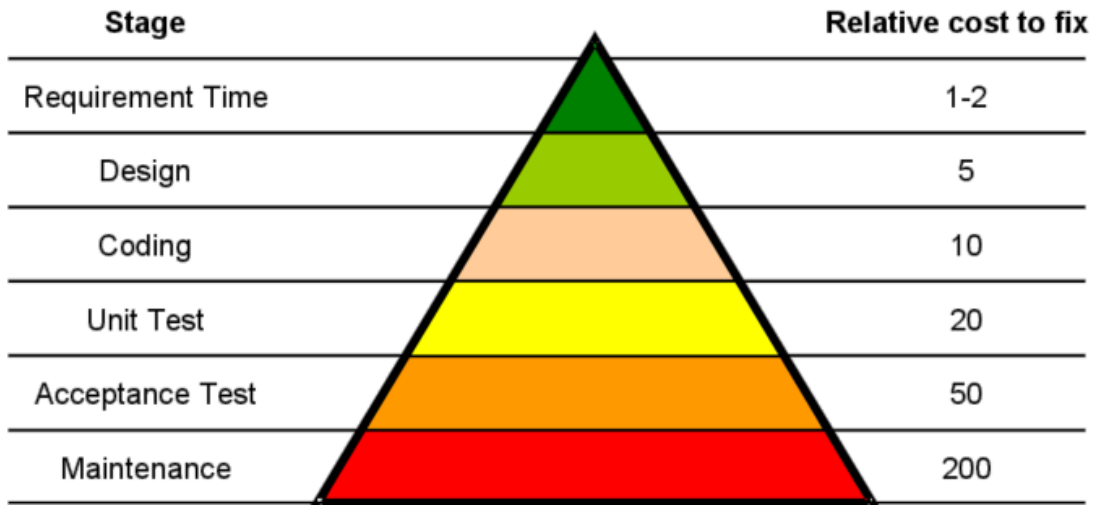
- Häufige Änderungen der Anforderungen / Probleme beim Änderungsmanagement
- Unvollständige, fehlende oder unzureichend definierte Anforderungen
- Unzureichende Einbeziehung der Interessengruppen (insbesondere der Endnutzer)
- Unrealistische Erwartungen
- Missverstandene Anforderungen
- Unrealistische Planung / fehlende Planung, unzureichende Ressourcen
- Mangelnde Unterstützung durch das Management
- Probleme bei der Kommunikation und Zusammenarbeit
- Unzureichende Tests

stehen im Zusammenhang mit Requirements Engineering

Bewertung der CHAOS-Studie

- Kritiker **zweifeln an** den Methoden und Ergebnissen
- **Die praktische Erfahrung bestätigt** jedoch, dass
  - **alle** Beteiligten **von Anfang an** einbezogen werden sollten
  - Anforderungsanalyse und Softwarearchitektur einen **entscheidenden Einfluss** auf den Erfolg eines Projekts haben

- Einfache Erkenntnis: "Je früher ein Fehler gefunden wird, desto einfacher und damit billiger ist er zu beheben."
- Faustregel: Wenn die Behebung eines Fehlers im Anforderungsdokument 1 (Stunde/Tag/...) dauert, dauert es 200 (Stunden/Tage/...), wenn das System fertig ist und läuft.



Was ist eine Anforderung?

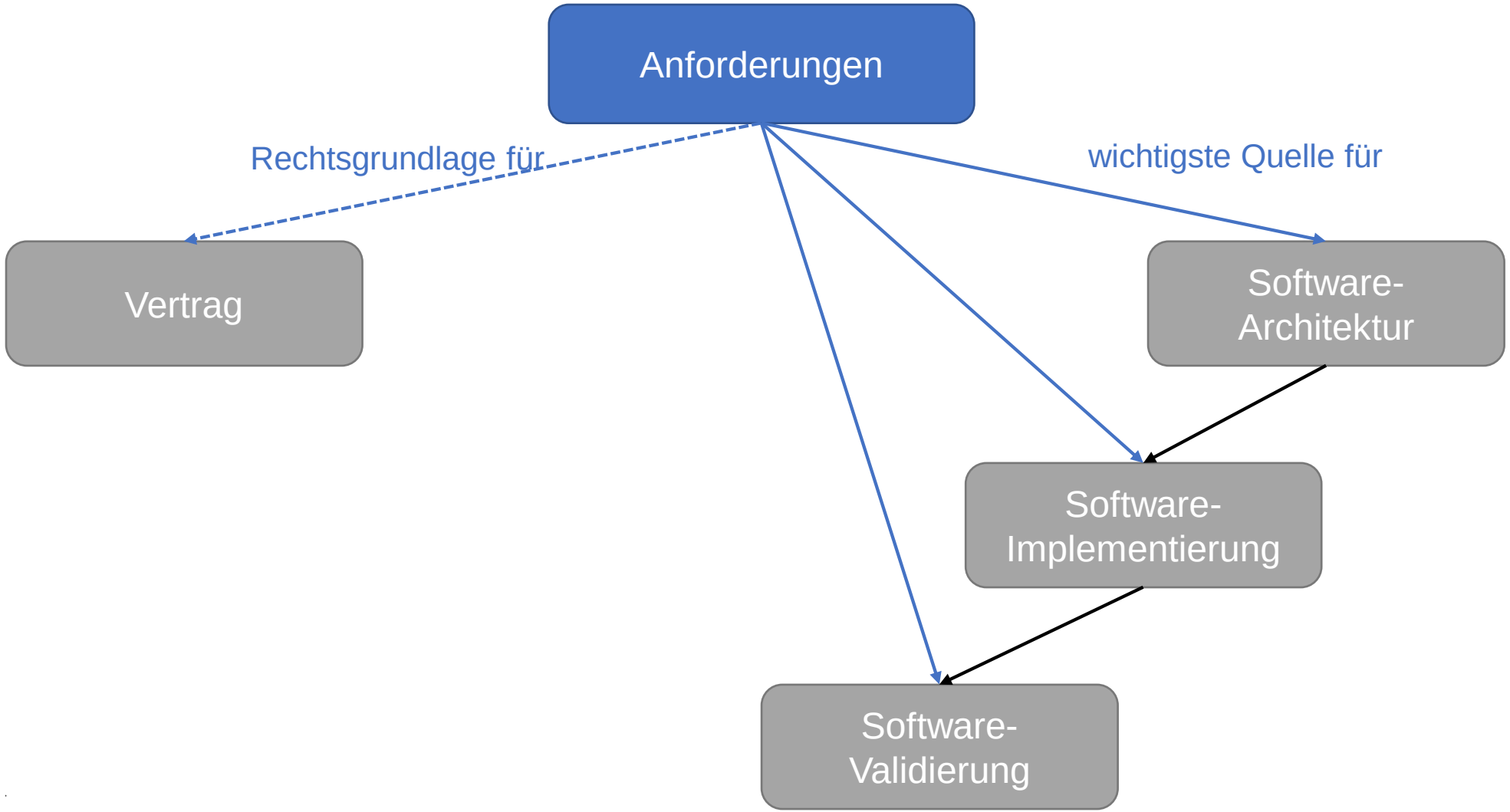
Aus dem IREB-Glossar:

- **Anforderung:**
  - Ein von einem Stakeholder wahrgenommenes Bedürfnis.
  - Eine Fähigkeit oder Eigenschaft, die ein System haben muss
  - Eine dokumentierte Darstellung eines Bedürfnisses, einer Fähigkeit oder einer Eigenschaft.
- **Stakeholder**: Eine Person oder Organisation, die (direkt oder indirekt) Einfluss auf die Anforderungen eines Systems hat oder von diesem System betroffen ist.

Anmerkung:

- Es gibt kein einheitliche Auffassung darüber, wie genau eine Anforderung formuliert werden muss.
- Sie kann von einer **abstrakten Beschreibung** eines Dienstes oder einer Systembeschränkung **bis hin zu einer detaillierten** mathematischen **Funktionsspezifikation reichen**.

Bedeutung der Anforderungen



Anforderungsspezifikation, größte Herausforderung 1

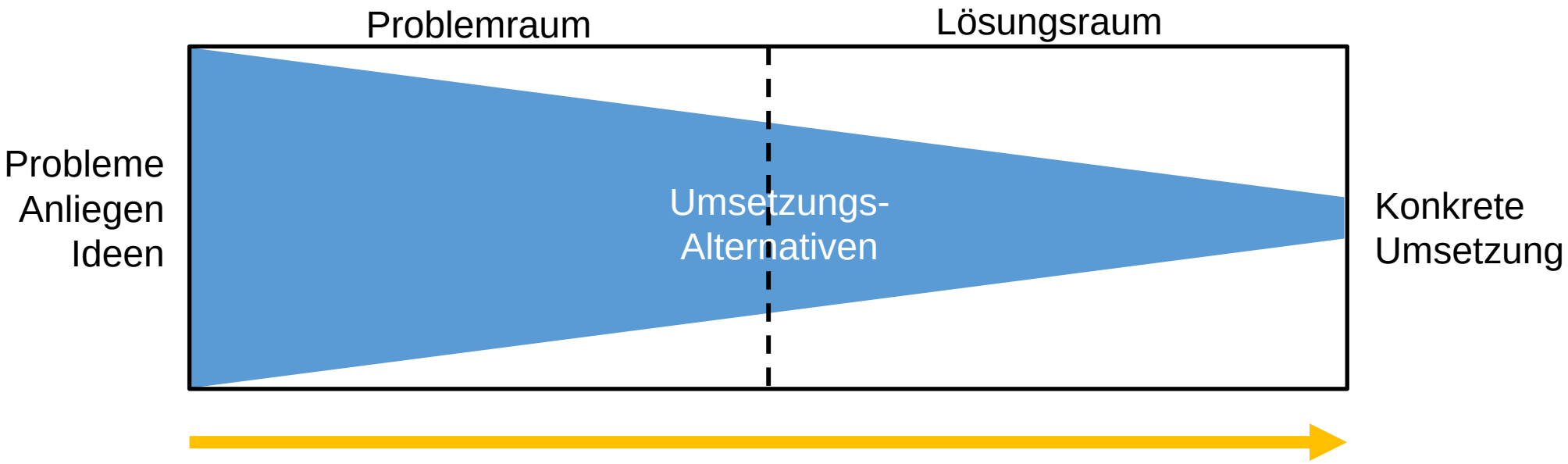
- Erreichen von **Vollständigkeit und Konsistenz**
  - Grundsätzlich sollten die Anforderungen sowohl vollständig als auch konsistent sein.
    - **Vollständig:** Sie sollten Beschreibungen aller Möglichkeiten enthalten.
    - **Widerspruchsfrei:** Es sollte keine Konflikte oder Widersprüche in den Beschreibungen der Systemeinrichtungen geben.
  - In der Praxis ist es aufgrund der Komplexität der Systeme und der Umwelt sowie der Anzahl der verschiedenen beteiligten Akteure kaum möglich, ein vollständiges und konsistentes Anforderungsdokument zu erstellen, sobald ein System eine gewisse Größe erreicht hat.

Anforderungsspezifikation, größte Herausforderung 2

- **Verstehen, was der Nutzer will**
  - "Nutzer sagt A, meint B, will C"
  - *YOU ARE NOT THE USER!*
    "**Sie** (der Entwickler/Architekt/...) **sind nicht Benutzer:in!** ", das sollten Sie immer im Hinterkopf behalten
  - Anforderungen können sich ändern, daher
    - Verifizierung der Anforderungen so oft wie möglich (Prototypen, Software-Inkremente, Demos)
  - Benutzer:in (oder Entwickler:in) **schlägt bereits Lösungen** und **Tools vor**
    - Bleiben Sie so lange wie möglich in dem **Problembereich**
    - Versuchen Sie, über den Tellerrand zu **schauen** und kreative Lösungen zu finden

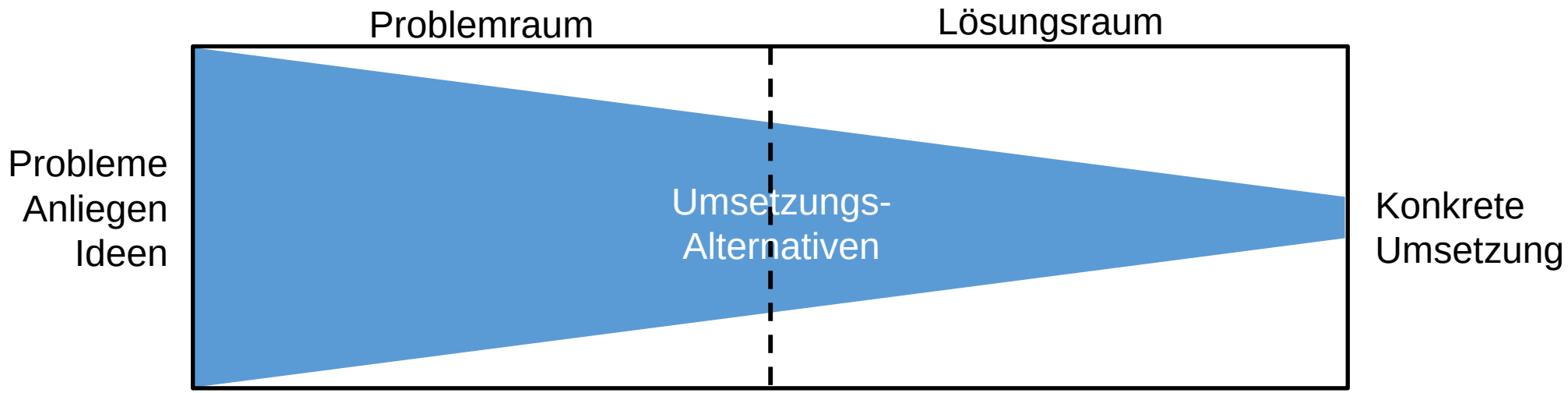
*"Ich brauche einen **Hammer**, um die **Nägel** einzuschlagen!"*
***Aber: Schrauben** wären die viel geeignetere Lösung!*

Problemraum, Lösungsraum



– Beim Software-Engineering geht es darum, den Problem- und Lösungsraum einzugrenzen, um eine angemessene Lösung zu finden, die den Anliegen aller Beteiligten gerecht wird.

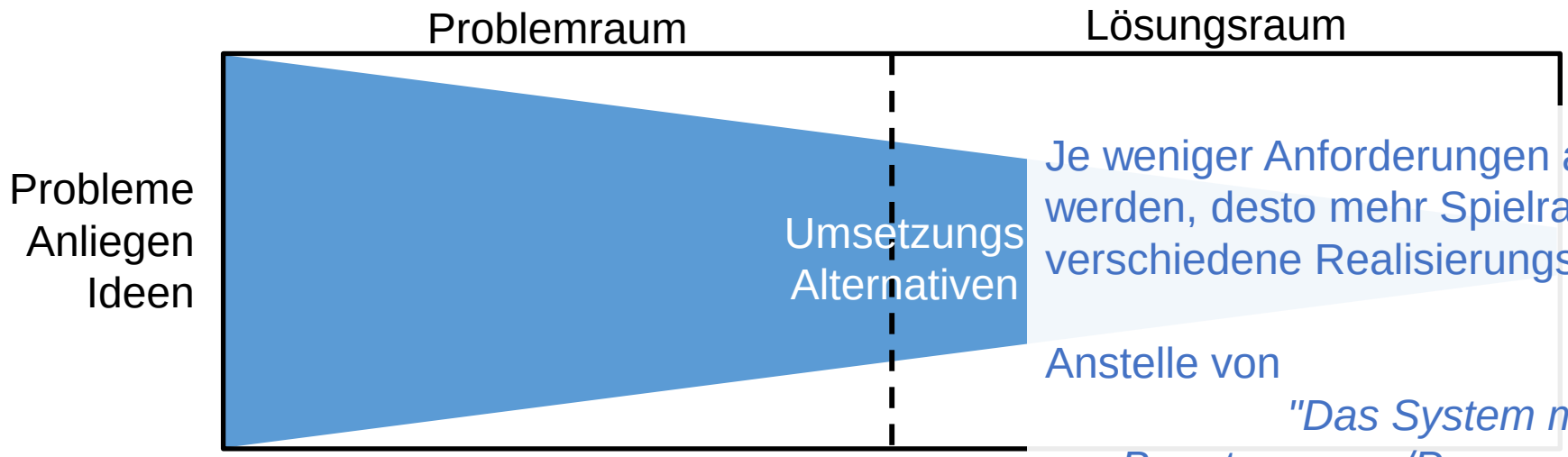
Benutzer- und Systemanforderungen



- Anforderungen, die den Anwendungskontext und die Funktionalität eines Systems beschreiben
- Nutzerperspektive, "Benutzeranforderungen"
- *"Ein Benutzer muss sich am System authentifizieren."*
- Hilfe bei der Klärung der Funktionalität und des Systemumfangs

- Anforderungen, die bestimmte Eigenschaften des Systems beschreiben
- Systemperspektive, "Systemanforderungen"
- *"Das System muss einen Benutzernamen/Passwort-Dialog anbieten".*
- das System einzuschränken und zu präzisieren

Wie genau soll man sein?



- Anforderungen, die den Anwendungskontext und die Funktionalität eines Systems beschreiben
- Nutzerperspektive, "Benutzeranforderungen"
- *"Ein Benutzer muss sich am System authentifizieren."*
- Hilfe bei der Klärung der Funktionalität und des Systemumfangs

Je weniger Anforderungen an das System gestellt werden, desto mehr Spielraum gibt es für verschiedene Realisierungsalternativen.

Anstelle von

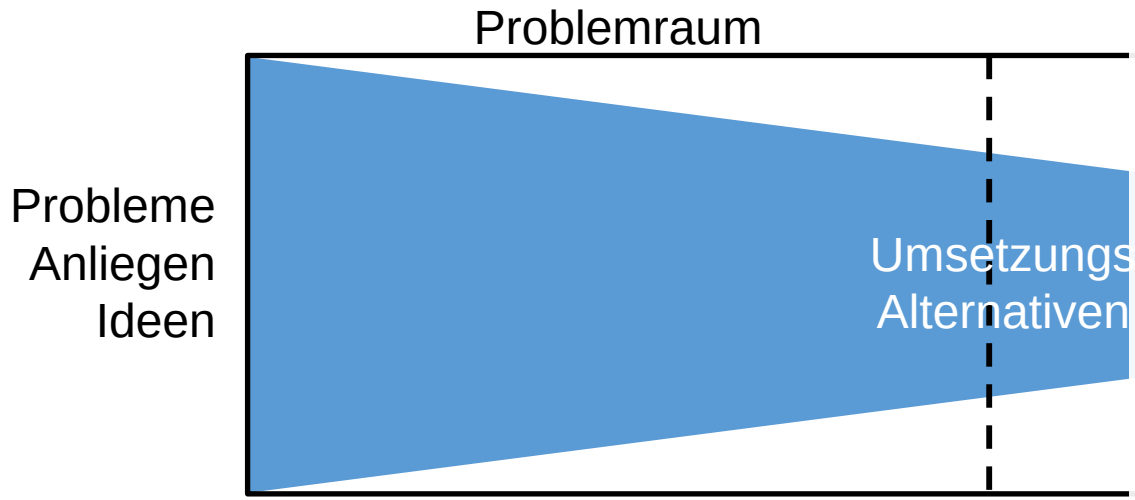
"Das System muss einen Benutzername/Passwort-Dialog anbieten".

könnte die Lösung durch

"Das System schickt dem Benutzer einen Anmeldecode per SMS".

für die gegebenen Nutzeranforderungen (z. B. im Rahmen einer mobilen Anwendung) die wesentlich geeignetere Lösung sein.

Wie genau soll man sein?



- Anforderungen, die den Anwendungskontext und die Funktionalität eines Systems beschreiben
- Nutzerperspektive, "Benutzeranforderungen"
- *"Ein Benutzer muss sich am System authentifizieren."*
- Hilfe bei der Klärung der Funktionalität und des Systemumfangs

Bei **kundenspezifischen Produkten** ("Individual-Software") kann der Kunde die **Benutzeranforderungen** als **Grundlage für ein Vertragsangebot** angeben, während die Bieter (Softwareunternehmen) dann die Grundlage für den **eigentlichen Vertrag** bilden können, indem sie die **Systemanforderungen** für die Benutzeranforderungen spezifizieren.

Dennoch gibt es in der Regel auch Systemanforderungen, die bereits vom Kunden vorgegeben sind, z.B. die Verwendung bestimmter Technologien, die im Unternehmen im Einsatz sind.
Diese werden in der Regel als **Randbedingungen (*Constraints*)** formuliert**.**

Requirements abstraction (Davis 1993)

“If a company wishes to let a contract for a large software development project, it must define its needs in a **sufficiently abstract way that a solution is not pre-defined**. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organization’s needs.
Once a contract has been awarded, the contractor must write a system definition for the client **in more detail so that the client understands and can validate** what the software will do.
Both of these documents may be called the requirements document for the system."

Alan M. Davis, *Software Requirements: Objects, Functions and States* (Prentice Hall, 1993)

Requirements abstraction (Davis 1993)

Übersetzt: "Wenn ein Unternehmen einen Auftrag für ein großes Softwareentwicklungsprojekt vergeben will, muss es seinen Bedarf **so abstrakt** definieren**, dass eine Lösung nicht vorgegeben ist**. Die Anforderungen müssen so formuliert sein, dass sich mehrere Auftragnehmer um den Auftrag bewerben können, die (unter Umständen) verschiedene Möglichkeiten anbieten, um die Anforderungen des Kundenunternehmens zu erfüllen.
Nach der Auftragsvergabe muss der Auftragnehmer eine **detaillierte** Systemdefinition für den Kunden verfassen**, damit der Kunde versteht,** was die Software leisten wird**, und dies auch überprüfen kann**.
Diese beiden Dokumente können als Anforderungsdokument für das System bezeichnet werden."

- "Lastenheft"
- "Pflichtenheft"

Alan M. Davis, *Software Requirements: Objects, Functions and States* (Prentice Hall, 1993)

Funktionale und nicht-funktionale Anforderungen

- **Funktionale Anforderungen**
  - Eine Anforderung bezüglich eines Ergebnisses oder Verhaltens, das von einer Funktion eines Systems bereitgestellt werden soll. (IREB-Glossar)
- **Nicht-funktionale Anforderungen**
  - Jede Anforderung, die keine funktionale Anforderung* ist, wie z. B. Qualitätsattribute, Geschäftsziele und Randbedingungen/Einschränkungen (*Constraints*)
  - **Sie beziehen sich** oft **auf das System als Ganzes** und nicht auf einzelne Funktionen oder Dienste.

* Es gibt verschiedene Definitionen für nicht-funktionale Anforderungen, die restriktiver sind und nur (bestimmte) Qualitätsattribute zählen. In diesen Definitionen ist ein Constraint und ein Geschäftsziel eine andere Art von Anforderung.

Funktionale Anforderungen

- Eine Anforderung bezüglich eines Ergebnisses oder Verhaltens, das von einer **Funktion** eines Systems bereitgestellt werden soll.
- Kann auch angeben, was das System **nicht** tun soll.

Mögliche Aspekte:

- **Funktionen**: Eingabe, Verarbeitung, Ausgabe
- **Daten**: Aufbau, Nutzung, Erzeugung, Speicherung, Übertragung, Veränderung.
- **Verhalten**: Sichtbares dynamisches Systemverhalten, Interaktion der Funktionen (untereinander und mit den Daten)
- **Fehlerfälle**: Welche Fälle sind unerwartet? Wie behandelt das System diese Fälle?

Nicht-funktionale Anforderungen

- **Geschäftsziele**, z. B. eine weltweite Einführung der Software
- **Qualitätsattribute zur Laufzeit**, z.B. Zuverlässigkeit, Reaktionszeit und Speicherbedarf
- **Qualitätsattribute zur Entwicklungszeit**, z.B. Testbarkeit der Benutzeroberfläche mit automatisierten Tests
- **Technische Randbedingungen**, z. B. eine bestimmte verfügbare Hardware mit bestimmter I/O-Performance oder vorhandene Middleware, auf der das System laufen soll (z. B. WebLogic Application Server, MySQL-Datenbank)
- **Rechtliche Randbedingungen**, z. B. die Auswirkungen der Datenschutz-Grundverordnung (DSGVO/GSPR), wonach ein Nutzer das Recht hat, alle seine Daten aus dem System zu löschen
- **Andere Randbedingungen** (Bereich, Organisation, Kosten und Zeit)
- **Nicht-funktionale Anforderungen können kritischer sein als funktionale Anforderungen.** Wenn diese nicht erfüllt werden, kann das System unbrauchbar sein.
  - An dieser Stelle kommt die Software-Architektur ins Spiel!
  - Mehr dazu später (**Architektur-Treiber**)

Wie kann man zwischen funktionalen und nicht-funktionalen Anforderungen unterscheiden?

- **Es ist nicht immer klar, ob eine Anforderung rein funktional oder nicht-funktional ist!**
- Beispiel: "Das System ist in der Lage, falsche Benutzereingaben zu erkennen."
  - Funktional?
    - "Bei falscher Eingabe Fehlermeldung anzeigen".
    - Anzeige einer Fehlermeldung ist Funktion!
  - Nicht-funktional?
    - "Das System stürzt bei falschen Daten nicht ab."
    - Qualitätsmerkmal *Robustheit*
- Einteilung "funktionale Anforderungen und nicht-funktionale Anforderungen" **taugt nicht** als Hauptebene der Anforderungsklassifizierung.
- Besser alle Anforderungen (funktional und nicht-funktional) zusammennehmen, die ein gemeinsames Thema betreffen, z.B.
  - Benutzerregistrierung
  - Benutzeranmeldung
  - Datensicherung, …

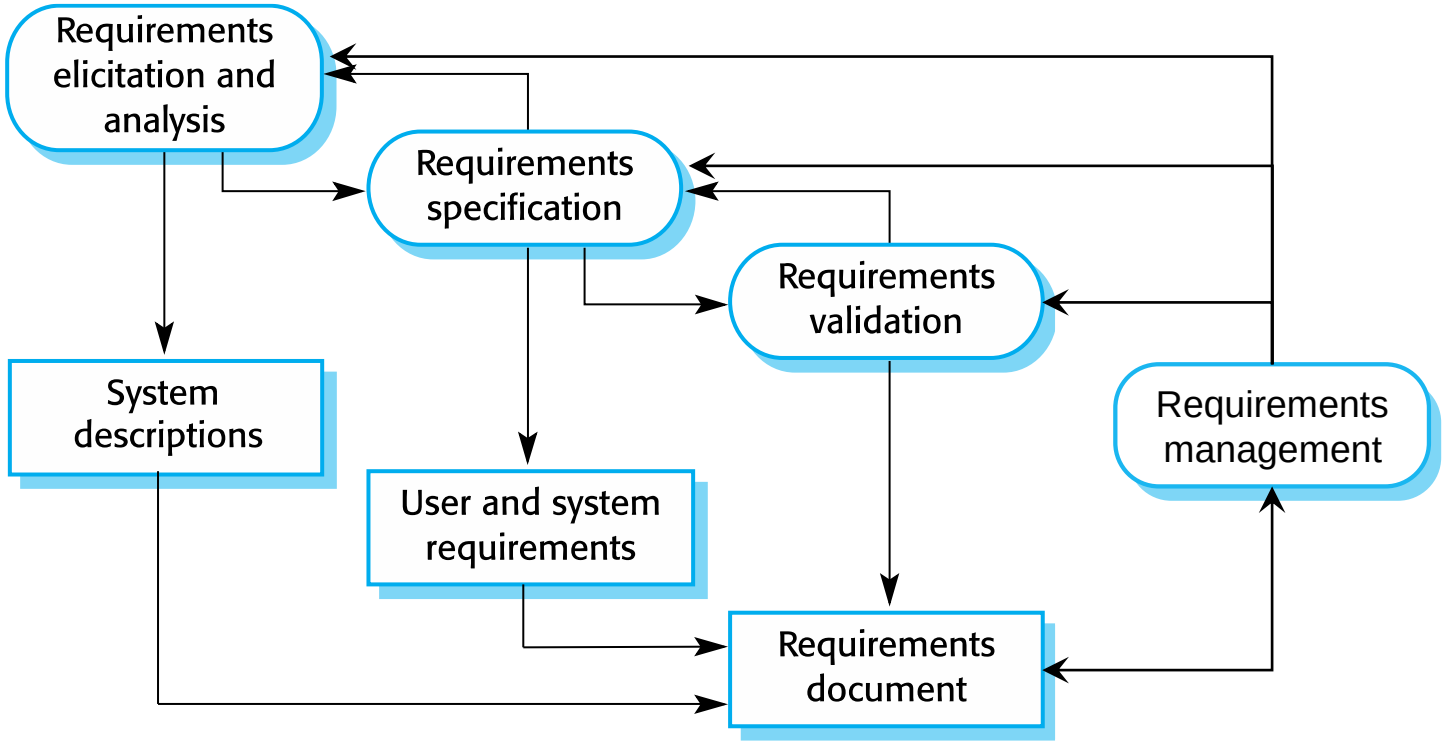
Requirements Engineering (RE)

Aus dem IREB-Glossar (übersetzt):

- **Requirements Engineering** ist der **systematische** und **disziplinierte** Ansatz zur **Spezifikation** und **Verwaltung** von **Anforderungen** mit dem Ziel
  - **die** Wünsche und Bedürfnisse der **Beteiligten zu verstehen** und
  - das **Risiko minimieren**, ein System zu liefern, das diese Wünschen und Bedürfnissen nicht erfüllt.

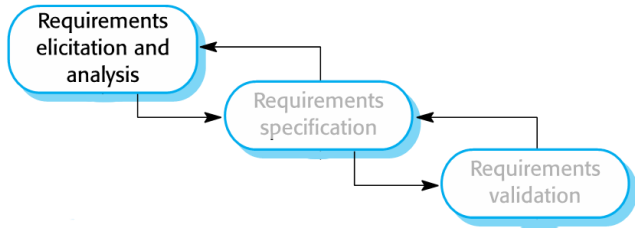
RE-Aktivitäten

- **Anforderungserhebung** – *Ermitteln*
  - Ermittlung, Sammlung und Konsolidierung von Anforderungen aus verschiedenen Quellen
- **Anforderungsspezifikation** – *Dokumentieren*
  - Systematische Darstellung und Dokumentation der ermittelten Anforderungen
- **Validierung von Anforderungen** – *Validieren*
  - Überprüfung, ob die dokumentierten Anforderungen den tatsächlichen Bedürfnissen/Zielen der Beteiligten entsprechen
  - Kann Verhandlungen zwischen verschiedenen Interessengruppen nötig machen
- **Anforderungsmanagement** - *Verwalten*
  - Verwaltung bestehender Anforderungen und zugehöriger Artefakte.



Erweiterung einer Abblildung von Ian Sommerville

Kennen Sie Ihre Stakeholder!



- **Stakeholder** *(Interessensgruppen):*
  - Jede Person oder Organisation, die in irgendeiner Weise von dem System betroffen ist und somit ein berechtigtes Interesse hat (Ian Sommerville)
  - Eine Person oder Organisation, die (direkt oder indirekt) Einfluss auf die Anforderungen eines Systems hat oder die von diesem System betroffen ist. (IREB)
- **Stakeholder-Typen**
  - Endbenutzer:innen (*End users*)
  - Systemverantwortliche (*System managers*)
  - Systembesitzer:in (*System owners*)
  - Systementwicklungsteam (*System development team*)
  - Systemtest-, Betriebs- und Support-Teams (*system testing, operation and support teams*)
  - Interne und externe Stakeholder
    - der Umfang (*Scope*) von intern/extern kann variieren
- Ein wichtiger Teil des Projektmanagements besteht darin, **alle relevanten Stakeholder zu identifizieren**.
- **Von dort kommen die Anforderungen!**

Beispiel

Beispiel System: Digitale Dörfer Plattform

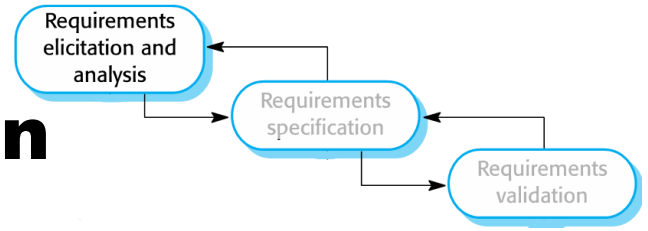
- https://www.youtube.com/watch?v=uNOP4R-SsxY (Englisch)
- https://www.youtube.com/watch?v=fHYpKpZzklo (Deutsch)

Stakeholder Digitale Dörfer

Beispiel

- Fraunhofer IESE
  - Projektleitung
  - Team für Entwicklung, Tests und Betrieb (DevOps)
  - Support-Team
  - Institutsleitung (Geldgeber und strategische Ausrichtung)
- Innenministerium Rheinland-Pfalz (Geldgeber)
- Entwicklungsagentur Rheinland-Pfalz (Geldgeber)
- Kommunen Eisenberg, Göllheim und Betzdorf-Gebhardsheim (Projektpartner, Modellregion)
  - Bürgermeister
  - Verwaltungspersonal
  - Bewohner:innen

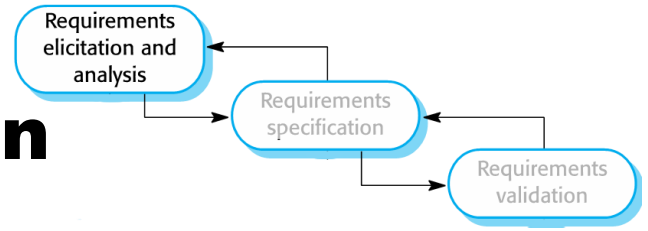
Methoden zur Ermittlung von Anforderungen



- **Interviews/Umfragen**
  - Interviews sind gut geeignet, um ein **Gesamtverständnis** dafür zu erlangen, was die Stakeholder tun und wie sie mit dem System interagieren können.
  - Die Interviewer:innen müssen **aufgeschlossen** sein **und dürfen keine vorgefassten Meinungen** darüber haben, was das System tun sollte.
  - Man bringt die Diskussion in Gang: mit einer **Eröffnungsfrage**, einem **Anforderungsvorschlag** oder durch die **gemeinsame Arbeit an einem Systemprototyp oder bestehenden System**

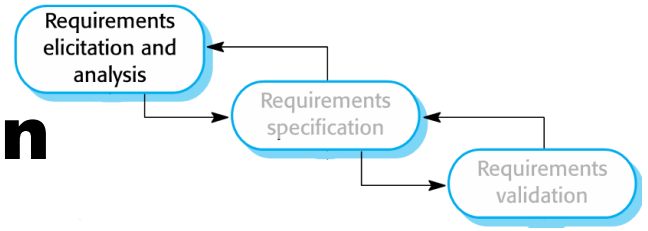
  - Finden Sie die Personen, die **den wertvollsten Beitrag** leisten können!
    - Es ist nicht immer einfach, diejenigen Personen zu finden, die wirklich die Hauptanforderungen stellen, oder Zugang zu ihnen zu erhalten.

Methoden zur Ermittlung von Anforderungen



- **Benutzertests/Walkthroughs**
  - **Man beobachtet** die Nutzer bei ihrer Arbeit, entweder
    - ohne die Software (Digitalisierungsprojekt) oder
    - mit der bisherigen Software (Neuentwicklung) oder
    - mit der aktuellen Software, die weiter entwickelt/verbessert werden soll
  - Man sollte ganz verschiedene Stakeholder einbeziehen, z.B. auch die IT-Mitarbeiter:innen, welche die Software warten oder testen

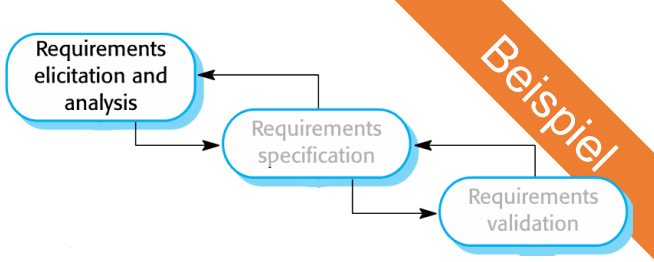
Methoden zur Ermittlung von Anforderungen



- **User Journeys (**auch **User Stories** genannt**)**:
  - **Beispiele aus dem wirklichen Leben**, wie die Software zur Erfüllung einer bestimmten Aufgabe eingesetzt werden kann
  - geschrieben aus der **Perspektive eines Nutzers**
  - Da sie auf einer konkreten Praxissituation basieren, **können sich die Beteiligten mit ihnen identifizieren** und sie aus Ihrer Sicht kommentieren.

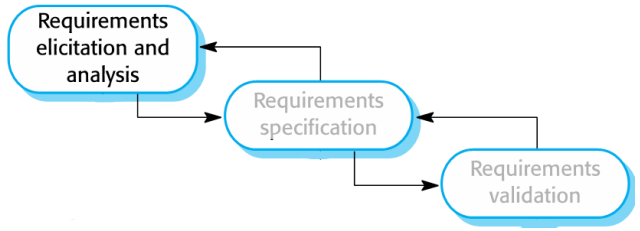
- Aufgepasst:
  - Bei **SCRUM** (agile Entwicklung) werden **die** Anforderungen in Form von so genannten **User Stories**
  - Diese sind nicht zu verwechseln mit User Journeys/Stories als Methode **der Anforderungsermittlung**, die in der Regel viel länger und aufwändiger sind.

Anforderungserhebung (DorfFunk)



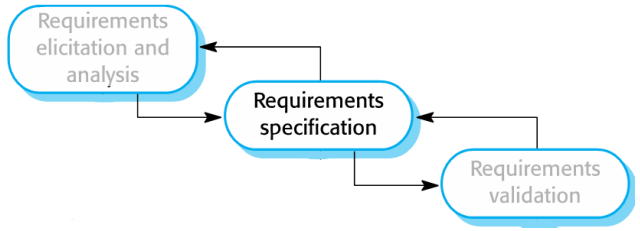
- Interview mit Peter, einem 45-jährigen DorfFunk-Poweruser
- Eröffnungsfrage: "Welche Funktionen vermissen Sie?"
- Antworten:
  1. "Ich möchte DorfFunk-Posts auf WhatsApp teilen."
  2. "Ich möchte mein Profilbild in der App aktualisieren können."
  3. "Eine **Suchfunktion** wäre schön. Ich habe Schwierigkeiten, ältere Beiträge zu finden."
- Beobachtungen:
  - Punkt 1 ist teuer in der Umsetzung.
  - Punkt 2 ist eine Funktion, die bereits vorhanden ist und daher leichter zugänglich gemacht werden sollte
  - Punkt 3 hat eine angemessene Komplexität und kann daher für die Umsetzung in der nächsten aufgenommen werden.

Probleme der Anforderungserhebung



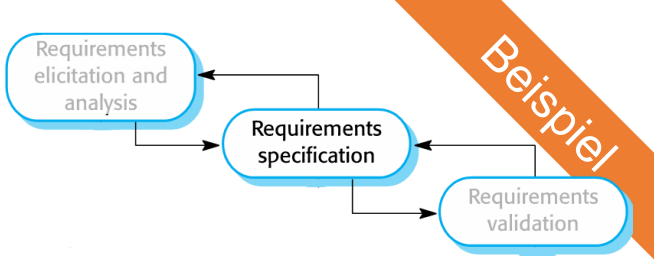
- *“Stakeholders typically know what they want, but not what they really need!”* [Davis, 1993] *(Stakeholder wissen in der Regel, was sie wollen, aber nicht, was sie wirklich brauchen!)*
- Die Stakeholder **drücken** ihre Anforderungen in ihren **eigenen Worten** aus.
  - Anforderungen müssen in eine Sprache "übersetzt" werden, die **allen Anforderungen gemeinsam** ist und **von allen Personen**, die die Software realisieren, **verstanden** wird.
- Verschiedene Interessengruppen können **widersprüchliche Anforderungen** haben.
  - Verhandlungsgeschick ist gefragt
- **Organisatorische und politische Faktoren** können die Systemanforderungen beeinflussen.
  - Manchmal muss man eine eigentlich überlegene Lösung verwerfen, auch wenn diese bessere Benutzerfreundlichkeit/technologische Vorteile/... hat
- Die **Anforderungen ändern sich** während des Erhebungsprozesses. Es können neue Stakeholder auftauchen und das Geschäftsumfeld kann sich ändern.
  - **Anforderungsmanagement** hilft, den Überblick über sich ändernde Anforderungen zu behalten

Darstellungen der Systemanforderungen Spezifikation



Notation	Beschreibung
Natürliche Sprache	Die Anforderungen werden in (nummerierten) **Sätzen in natürlicher Sprache** verfasst. Jeder Satz sollte eine Anforderung ausdrücken.
Strukturierte natürliche Sprache	Die Anforderungen werden in natürlicher Sprache in ein **Standardformular oder eine Vorlage** geschrieben. Jedes Feld enthält Informationen über einen Aspekt der Anforderung.
Entwurfsbeschreibungs-sprachen	Bei diesem Ansatz wird eine programmiersprachenähnliche Sprache verwendet, jedoch mit abstrakteren Merkmalen, um die Anforderungen durch die Definition eines Betriebsmodells des Systems zu spezifizieren. Dieser Ansatz wird nur noch selten verwendet, obwohl er für Schnittstellenspezifikationen nützlich sein kann.
Grafische Notationen	Zur Definition der funktionalen Anforderungen an das System werden grafische Modelle verwendet, die durch Textkommentare ergänzt werden; häufig werden UML-Use-Case- und Sequenzdiagramme verwendet.
Mathematische Spezifikationen	Diese Notationen beruhen auf mathematischen Konzepten wie **endlichen Automaten oder Mengen**. Obwohl diese eindeutigen Spezifikationen die Mehrdeutigkeit in einem Anforderungsdokument reduzieren können, verstehen die meisten Kunden eine formale Spezifikation nicht. Sie können nicht überprüfen, ob die Spezifikation das darstellt, was sie wollen, und zögern, sie als Vertragsgrundlage zu akzeptieren.

DorfFunk: Natürliche Sprache



Oft werden Anfoderungsdokumente auf Englisch verfasst:

– "A user can enter a search term and gets a list of items that include the search term. These items can be news posts ("News"), user posts ("Plausch"), offers ("Biete") and seekings ("Suche")."

Übersetzt:

– "Ein Nutzer kann einen Suchbegriff eingeben und erhält eine Liste von Beiträgen, die den Suchbegriff enthalten. Diese Beiträge können Nachrichten ("News"), Nutzerbeiträge ("Plausch"), Angebote ("Biete") und Suchen ("Suche") sein."

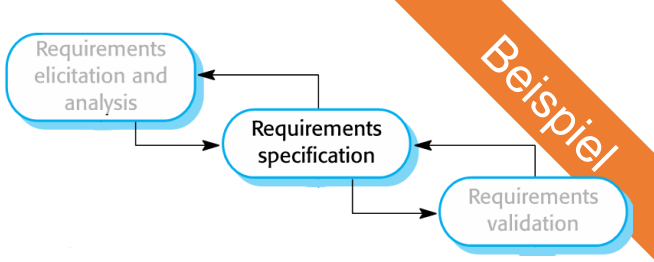
SCRUM: User-Stories

- *"Eine User Story ist eine informelle, allgemeine Erklärung einer Softwarefunktion, die aus der Perspektive des Endbenutzers geschrieben wird. Ihr Zweck ist es, darzulegen, welchen Wert eine Softwarefunktion dem Kunden bietet.*"
- Typischerweise in folgender Form aufgeschrieben
  - **Als [Nutzertyp] [möchte ich], [damit]". Englisch: As a [user type], I [want to], [so that]**
- "Als [Benutzertyp]": Für wen bauen wir das Produkt? Das Team sollte ein gemeinsames Verständnis davon haben, was dieser Nutzertyp ist, und idealerweise eine gewisse Empathie für diesen Nutzertyp entwickelt haben.
- "möchte ich": Hier beschreiben wir ihre Absicht - nicht die Funktionen, die sie verwenden. Was ist es, das sie tatsächlich erreichen wollen? Diese Aussage sollte frei von Implementierungsdetails sein (insbesondere sollte sie keine Teile der Benutzeroberfläche beschreiben).
- "damit": Wie passt ihr unmittelbarer Wunsch, etwas zu tun, in ihr Gesamtbild? Was ist der Gesamtnutzen, den sie zu erreichen versuchen? Was ist das große Problem, das gelöst werden muss?

Übersetzt vom Atlassian Agile Coach

- In der agilen Softwareentwicklung werden die Anforderungen oft in die Form von User Stories übertragen und manchmal sogar direkt als solche aufgeschrieben.

DorfFunk: User-Story



– As a DorfFunk power user, I want to find older entries that contain certain words, such that I can share these entries with other people instead of answering the same questions again and again."

übersetzt:

– "Als DorfFunk-Poweruser möchte ich ältere Einträge finden, die bestimmte Wörter enthalten, damit ich diese Einträge mit anderen Leuten teilen kann, anstatt immer wieder die gleichen Fragen zu beantworten."

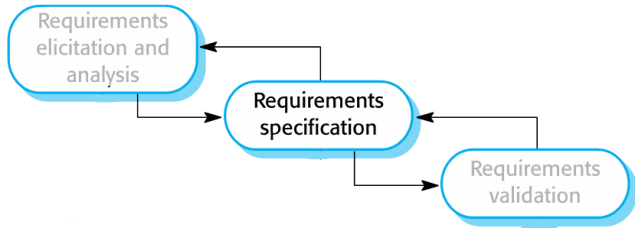
SCRUM: User stories → Aufgaben (Tasks)

- Aus einer User Story werden eine oder mehrere Implementierungsaufgaben abgeleitet
- Im Idealfall sollten die Geschichten den INVEST-Grundsätzen folgen:
  - **I**ndependent = Unabhängig
  - **N**egotiable = Verhandelbar
  - **V**aluable = Wertvoll
  - **E**stimable = Abschätzbar
  - **S**mall = Klein
  - **T**estable = Testbar

SCRUM: User stories → Aufgaben (Tasks)

- Aus einer User Story werden eine oder mehrere Implementierungsaufgaben abgeleitet
- Im Idealfall sollten die Geschichten den INVEST-Prinzipen folgen:
  - **I**ndependent = Unabhängig: Die Geschichten sollten in sich abgeschlossen sein und sich konzeptionell nicht überschneiden, so dass sie in beliebiger Reihenfolge geplant und durchgeführt werden können.
  - **N**egotiable = Verhandelbar: User Stories sind keine expliziten Verträge und sollten Raum für Diskussionen lassen.
  - **V**aluable = Wertvoll: Eine Story muss für die Stakeholder einen Wert haben.
  - **E**stimable = Abschätzbar: Der Umfang und der Implementierungsaufwand für eine Story sollten abschätzbar sein.
  - **S**mall = Klein: Eine Story sollte innerhalb der für ein Inkrement (=nächste Version) benötigten Zeit realisierbar sein wird
  - **T**estable = Testbar: Die Story oder die zugehörige Beschreibung muss die notwendigen Informationen liefern, damit Akzeptanztests spezifiziert werden können.
- In der Praxis können nicht alle Prinzipien erfüllt werden, z. B. gibt es meist Aufgaben, die von anderen Aufgaben abhängen, ...

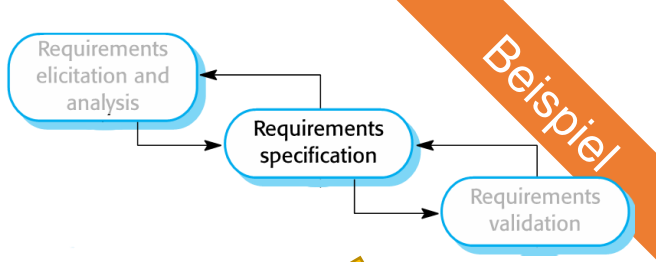
Given – When – Then



- Eine sehr verbreitete Form der Anforderungsspezifikation in **strukturierter natürlicher Sprache**
- Häufig für **Systemanforderungen** verwendet

- **Given** (Gegeben):
  - Beschreibt den Benutzer (Typ), die **Umgebung** und die Ausgangssituation
- **When** (Wenn):
  - Beschreibt **Stimulus** (Auslöser/Reiz), der eine Aktion oder ein Ereignis auslöst
- **Then** (Dann):
  - Beschreibt die erwartete **Reaktion** des Systems

Spezifikation (DorfFunk)



Für die Rückverfolgbarkeit bei der späteren Anforderungsvalidierung ist es wichtig, auf die Quelle einer Anforderung zu verweisen

Titel: Suche nach Beiträgen

ID: PostSearch

Quelle: Interview_Peter_2021-02-02.docx

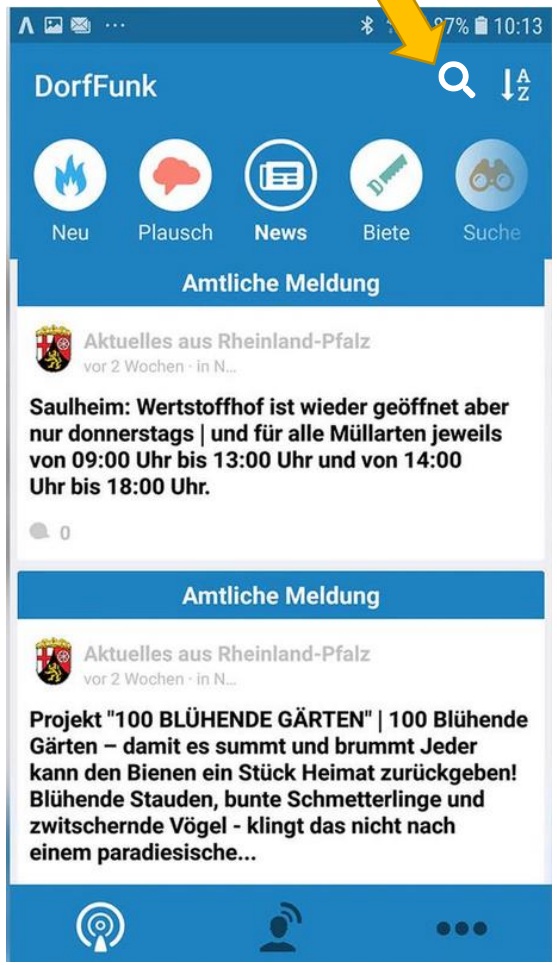
Given: Die DorfFunk-App wird gestartet und zeigt die **Liste der Nachrichten an**.

When: Der Nutzer klickt auf das Suchsymbol und gibt einen Suchbegriff ein.

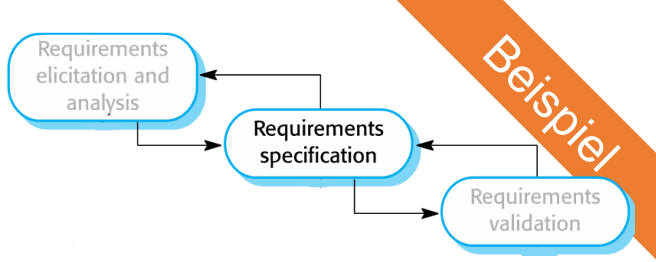
Then: Es werden nur Nachrichten aufgelistet, die den Suchbegriff enthalten.

Anmerkung: Diese Anforderung gilt auch, wenn der fettgedruckte Text ersetzt wird durch

- Liste der Benutzerbeiträge
- Liste der Suchanfragen
- Liste der Angebote



Spezifikation (DorfFunk)



Seien Sie präzise!

Titel: Suche nach Beiträgen

ID: PostSearch

Quelle: Interview_Peter_2021-02-02.docx

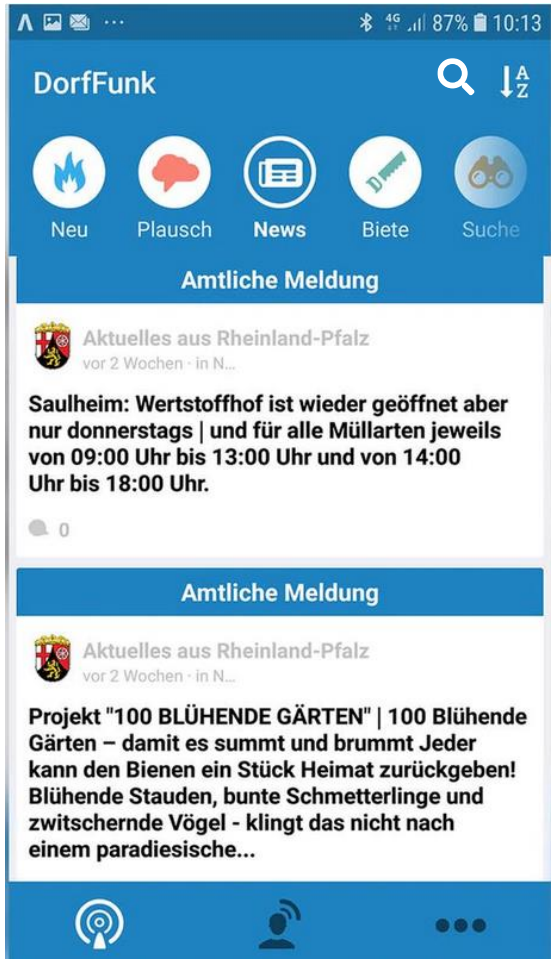
Given: Die DorfFunk-App wird gestartet und zeigt die **Liste der Nachrichten an**.

When: Der Nutzer klickt auf das Suchsymbol und gibt ***einen oder mehrere Begriffe*** ein.

Then: Es werden nur Nachrichten aufgelistet, die ***alle eingegebenen Wörter*** enthalten.

Anmerkung: Diese Anforderung gilt auch, wenn der fettgedruckte Text ersetzt wird durch

- Liste der Benutzerbeiträge
- Liste der Suchanfragen
- Liste der Angebote



Ableitung von Testfällen

Beispiel

Titel: Suche nach Beiträgen

ID: PostSearch

Given: Die DorfFunk-App wird gestartet und zeigt die **Liste der Nachrichten an**.

When: Der Nutzer klickt auf das Suchsymbol und gibt einen oder mehrere Begriffe ein.

Then: Es werden nur Nachrichten aufgelistet, die alle eingegebenen Wörter enthalten.

Anmerkung: Diese Anforderung gilt auch, wenn der fettgedruckte Text ersetzt wird durch

- Liste der Benutzerbeiträge
- Liste der Suchanfragen
- Liste der Angebote

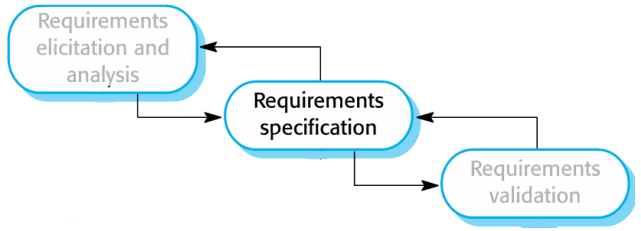
Diese Anforderung ist ein guter Ausgangspunkt für die Ableitung von Testfällen.

Verschiedene Testfälle:

- T_PostSearch_NewsItem_oneWord
- T_PostSearch_NewsItem_twoWords
- T_PostSearch_NewsItem_noEntryFound
- T_PostSearch_NewsItem_ManyEntriesFound
- T_PostSearch_UserPost_oneWord
- ...

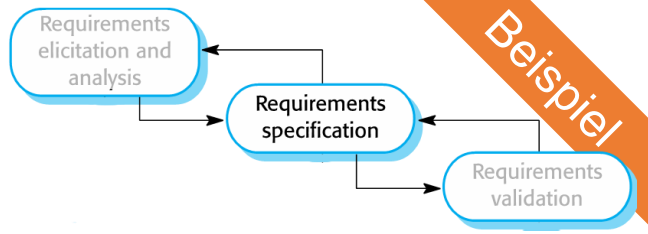
Diese Tests können manuell (durch Tester:in) oder sogar automatisch (automatisierte UI-Tests) durchgeführt werden.

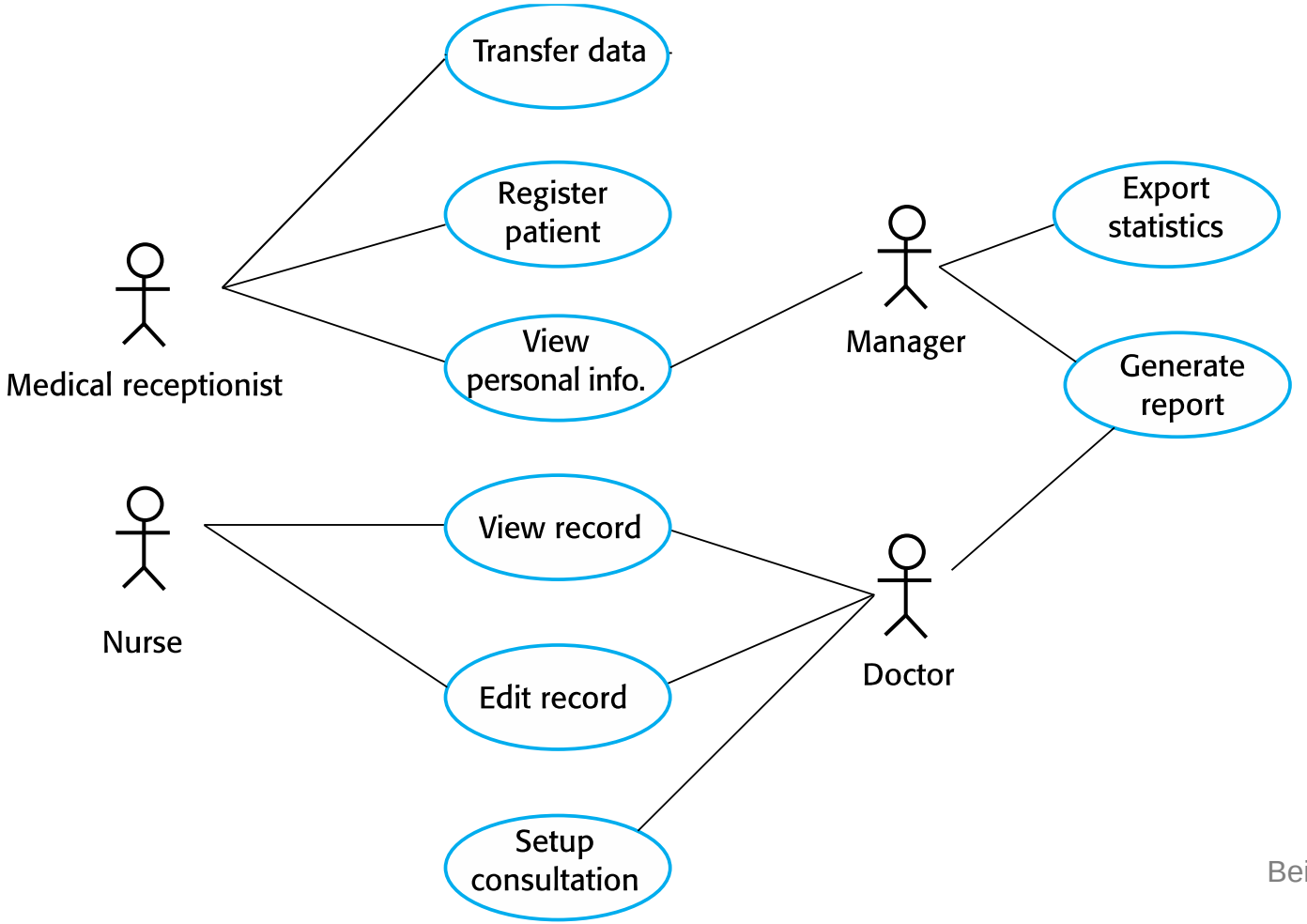
Use-Cases



- Use-Cases sind eine Art Szenario, das in der UML enthalten ist.
- Use-Cases identifizieren die Akteure in einer Interaktion und beschreiben die Interaktion selbst.
- Eine Menge von Use-Cases sollte alle möglichen Interaktionen mit dem System beschreiben.
- Es handelt sich um ein grafisches Modell auf hoher Ebene, ergänzt durch eine detailliertere tabellarische Beschreibung
- UML-Sequenzdiagramme können verwendet werden, um Details zu Anwendungsfällen hinzuzufügen, indem die Reihenfolge der Ereignisverarbeitung im System dargestellt wird.

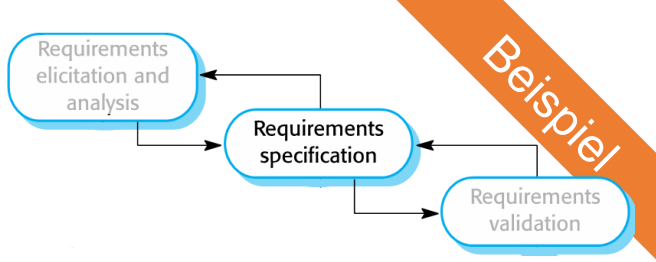
Use-Cases für ein medizinisches System





Beispiel von Ian Sommerville

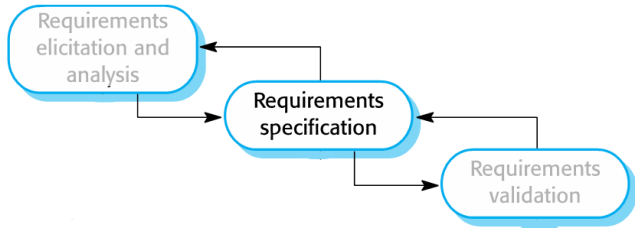
Tabular description of the ‘Transfer data’ use-case



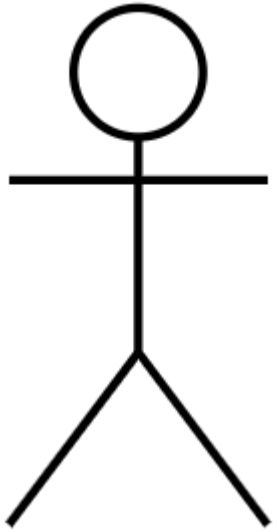
MHC-PMS: Transfer data	
Actors	Medical receptionist, patient records system (PRS)
Description	A receptionist may transfer data from the Mentcase system to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient’s diagnosis and treatment.
Data	Patient’s personal information, treatment summary
Stimulus	User command issued by medical receptionist
Response	Confirmation that PRS has been updated
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

Beispiel von Ian Sommerville

Benutzertypen



- Personen, die mit dem System interagieren, werden als **Endnutzer** oder **Systembenutzer** bezeichnet**.**
- Für die Spezifikation von Anwendungsfällen (und Anforderungen im Allgemeinen) ist es wichtig, **Typen von Endbenutzern zu identifizieren.**
- **Aufgepasst**: Ein Benutzer einer Kategorie kann verschiedene Rollen haben.
  - Z.B. kann ein "Student" ein "Übungsteilnehmer" oder "Tutor" sein
    - Das kann wichtig sein (App zur Übungsbewertung, → Modellierung als unterschiedliche Benutzertypen)
    - oder egal (App zur Immatrikulation → ein Typ "Student")

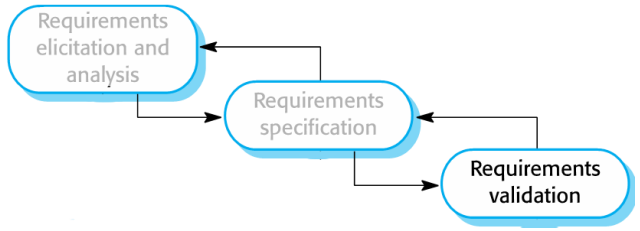


Seien Sie kritisch!

- Vergewissern Sie sich, dass jede Anforderung eine legitime Quelle hat!
- Stammen die Anforderungen wirklich vom Endnutzer?
  - Oder nur von jemandem, der *glaubt zu* wissen, was die Endnutzer:innen wollen?
- Sind die gegebenen Randbedingungen (Constraints) wirklich *hart und unveränderlich*?
- Welche Anforderungen lassen den größten Spielraum für Interpretationen?
  - Können sie verfeinert werden?
- Welches sind die Anforderungen, die den größten Einfluss auf das System haben?
  - Sind sie gut verstanden und dokumentiert?

Validieren Sie die Anforderungen!

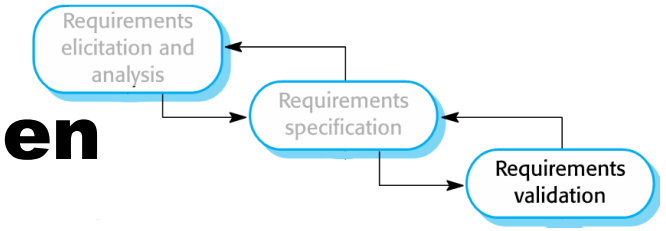
Validierung von Anforderungen



Zu prüfen:

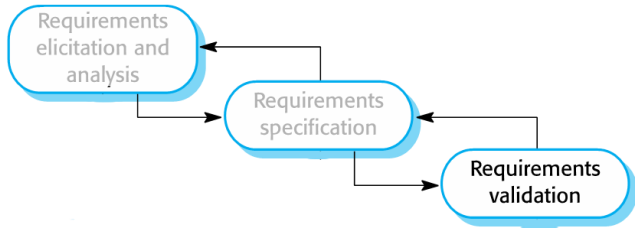
- ***Verifiability* – Überprüfbarkeit**
  - Ist die Anforderung realistisch überprüfbar?
- ***Comprehensibility* – Verständlichkeit**
  - Wurde die Anforderung richtig verstanden?
  - Können andere die Anforderung verstehen?
- ***Traceability* – Nachverfolgbarkeit**
  - Ist der Ursprung der Anforderung klar angegeben?
- ***Adaptability* – Anpassbarkeit**
  - Kann die Anforderung geändert werden, ohne große Auswirkungen auf andere Anforderungen zu haben?

Methoden zur Validierung von Anforderungen

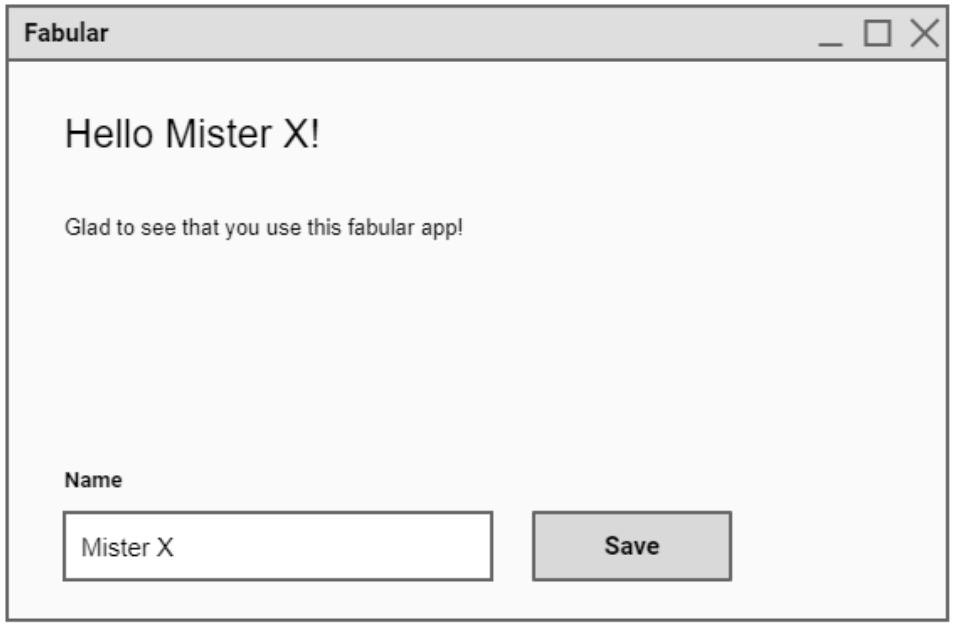


- **Bewertung der Anforderungen**
  - Systematische manuelle Durchsicht und Bewertung der Anforderungen.
- **Prototyping**
  - Verwendung eines ausführbaren Modells des Systems zur Überprüfung der Anforderungen.
- **Testfallerstellung**
  - Entwicklung von Tests für Anforderungen zur Überprüfung der Testbarkeit.

UI-Wireframes



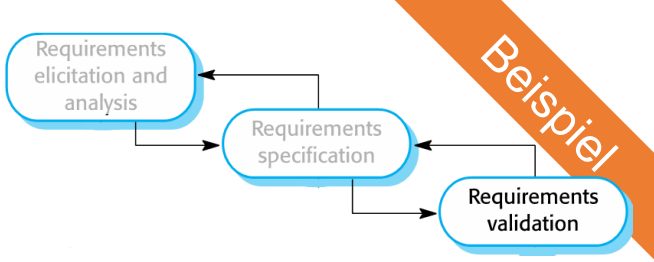
- Sonderfall des Prototyping
- Kann auf Papier oder mit Werkzeugunterstützung durchgeführt werden
- Leichte, kostengünstige Methode
- Unmittelbares Feedback des Nutzers



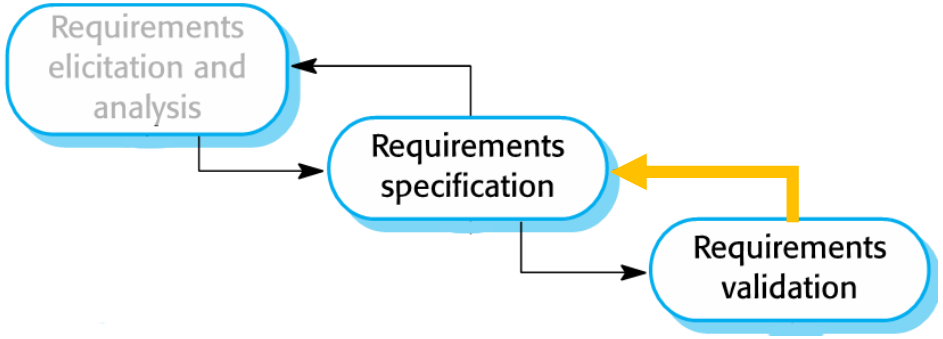
- Man konzentriert sich auf den Benutzerfluss oder das allgemeine Bildschirmlayout
- Man benutzt einfache, ungestylte Bildschirmansichten und nur selten Farben.

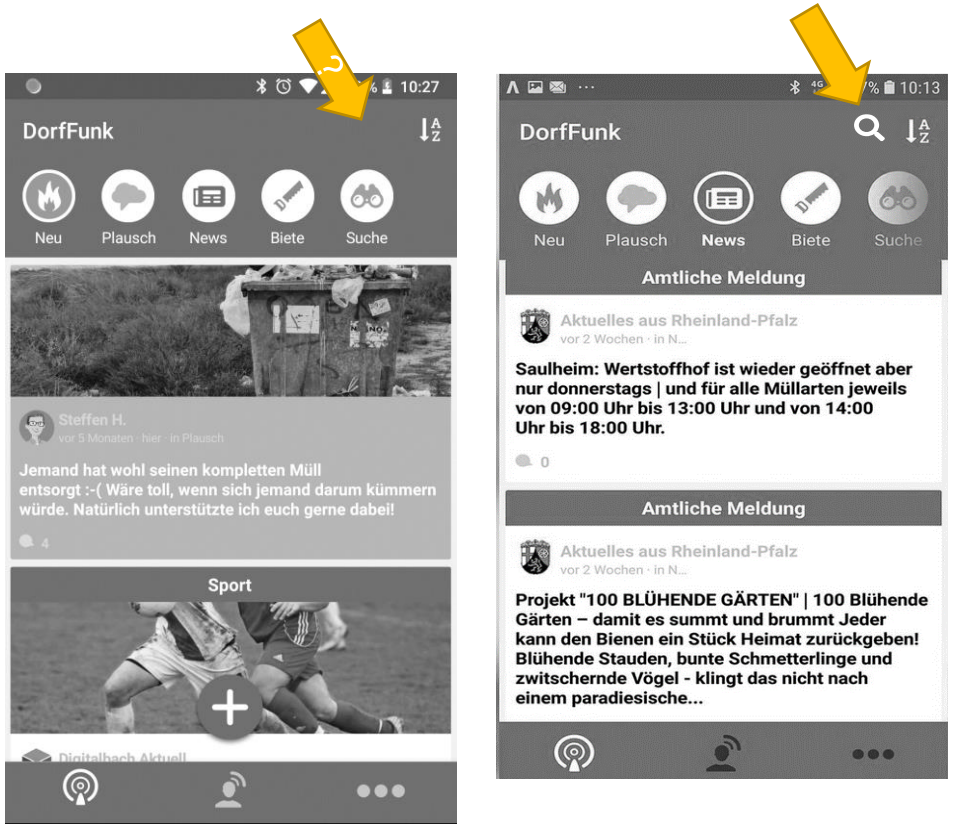
- Man gibt Testbenutzer:innen spezifische Aufgaben (z.B. "Ändern Sie Ihren Namen")
- Man erklärt, wenn eine bestimmte Funktionalität nicht modelliert wurde.
- Man darf dem Benutzer niemals genau sagen, was er tun soll. Man beobachtet und notiert jedes Verhalten, das sich von dem unterscheidet, was man erwartet hat. Man hilft nur dann weiter, wenn der Nutzer nicht mehr weiterweiß.

Validierung (DorfFunk)

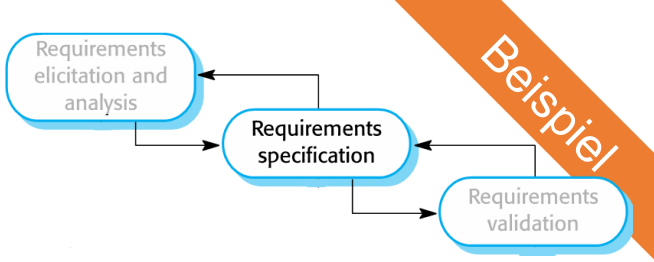


- Die UI-Wireframes für die Suchfunktionalität werden Peter gezeigt.
- Er stellt fest, dass die Suchschaltfläche **nur für jede Kategorie verfügbar** ist, aber **nicht als globale Suche.**
- Die Anforderung wurde **missverstanden** und nicht so spezifiziert, wie es der Stakeholder wollte
- Eine Überarbeitung ist erforderlich (neue Spezifikation)





Überarbeitete Spezifikation (DorfFunk)



Titel: **Globale** Suche nach Stellen

ID: PostSearch_v2

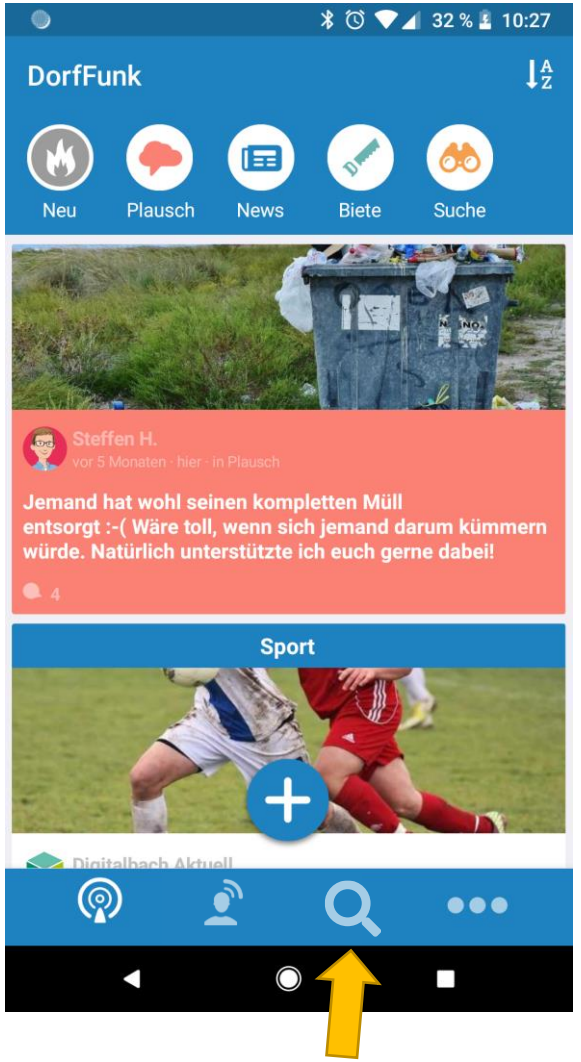
Quelle: Wireframe_Test_Peter_2021-03-15.docx

Given: Die DorfFunk-App wird gestartet und zeigt den Hauptbildschirm an.

When: Der Nutzer klickt auf das Suchsymbol und gibt einen oder mehrere Begriffe ein.

Then: Es werden nur Artikel aufgelistet, die alle eingegebenen Wörter enthalten. Diese Einträge werden aus

- Liste der Nachrichtenbeiträge
- Liste der Benutzerbeiträge
- Liste der Suchanfragen
- Liste der Angebote



Anforderungsmanagement

- Anforderungen kategorisieren
  - Rückverfolgbarkeit: Woher kommt die Anforderung? Quelle?
- Anforderungen zugänglich machen für alle Personen, die mit den Anforderungen arbeiten
  - Anforderungsdokument, und/oder
  - Tool-unterstützt (extra Typ in Issue-Tracker, Wiki, ..., spezialisierte Anforderungsmanagement-Tools)
- Anforderungen evaluieren
  - Sind die Anforderungen noch gültig?
  - Hat sich eine Anforderung geändert?

  → Management von Anforderungsänderungen