# Requirements-Engineering

Die SOPHISTen

## »Clever & kompakt«

# Die RE-Fibel von SOPHIST

SOPHIST

Die SOPHISTen

SOPHIST GmbH
Vordere Cramergasse 13
90478 Nürnberg
Deutschland
www.sophist.de

@ SOPHIST_GmbH
@SOPHIST.GmbH
/sophistgmbh
blog.sophist.de

1. Auflage 2021

Copy Editing & Herstellung: Roland Kluge, SOPHIST GmbH
Umschlaggestaltung und Layout: Heike Baumgärtner, Büro Hochweiss
Druck: Flyeralarm

# Requirements-Engineering

Die SOPHISTen

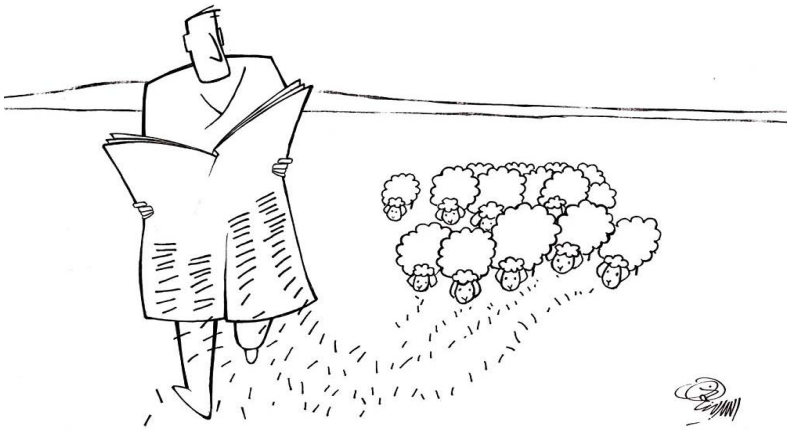
## »Clever & kompakt«

# Die RE-Fibel von SOPHIST

www.sophist.de

# 1. Einleitung und Motivation

Wir SOPHISTen sind seit über 20 Jahren als Berater und Trainer im Requirements-Engineering (RE) tätig. Dabei unterstützen wir Kunden aus den unterschiedlichsten Branchen, von der reinen Softwareentwicklung bei Versicherungen und Banken bis hin zur Durchführung von hardwareorientierten Systems-Engineering-Projekten im Automotivebereich oder in der Planung von Gebäudeanlagen. Der Schwerpunkt unserer Arbeit liegt dabei auf dem Ermitteln von Anforderungen, aus ihnen gute Systemanforderungen herzuleiten und diese zu verwalten. Weiterhin unterstützen wir bei dem Vermitteln der Anforderungen, je nach den Projektvorgaben, mittels einer Dokumentation der Anforderungen bis hin zu einem Storytelling. Aber wir beschäftigen uns auch mit weiteren Themen, die in einem engen Zusammenhang zum Requirements-Engineering stehen. So müssen wir das Thema Anforderungen in ein Systems-Engineering-Vorgehen einbetten, die Grundlage für Anforderungen mit Geschäftsprozessen bilden, oder auch Variantenbildung für ein Produkt bei den Anforderungen berücksichtigen. Aber auch aktuelle Themen (z.B. Smart Ecosystems, Digitale Transformation) und Strömungen (Videos im RE, Crowd-RE, ...) greifen wir in unseren Arbeiten auf. Darüber hinaus ist die Einführung des Requirements-Engineerings sowie die Einführung von agilen Vorgehensweisen Fokus unserer Tätigkeit.



Unser Leben ist stark von IT-Systemen geprägt – hängt teilweise sogar von ihnen ab. Sie machen unser Dasein deutlich angenehmer, strukturieren und gestalten es mit. Sie liefern Informationen, unterstützen bei Entscheidungen oder Arbeitsvorgängen und automatisieren Vorgänge. Und sie ermöglichen uns Dinge, von denen wir vor wenigen Jahren noch nicht mal geträumt haben. Wir leben in einem Smart Home, unser Auto besteht aus intelligenten Komponenten, unsere Haushaltsgeräte kommunizieren mit uns und das Smartphone verbindet uns immer und überall mit dem Rest der Welt. In der Industrie werden Waren smart gefertigt, die Landwirtschaft, die Energieversorgung, das Gesundheitssystem ... leisten mehr, da Systeme die

Wertschöpfung überwachen und optimieren. Damit all das zu einem Traum für die Menschheit wird und nicht im Albtraum endet, ist es wichtig, dass die Systeme das tun, was sie tun sollen und was wir von ihnen wollen. Genau hier setzt Requirements-Engineering an. Diese immer komplexeren Leistungen, die Systeme übernehmen, müssen erfunden oder ermittelt, analysiert und vermittelt werden und das entstandene Wissen (die Anforderungen) muss dann oft auch dokumentiert und verwaltet werden. Vorher müssen Sie die vom System zu unterstützenden Geschäftsprozesse verstehen und skizzieren. Die Kunst dabei ist es, auf die mannigfaltig vorliegenden Rahmenbedingungen einzugehen. Requirements-Engineering bedeutet hier die richtige Form für den richtigen Zweck zu finden.

Um Ihnen die graue Theorie ein bisschen farbiger zu gestalten, und die hier angesprochenen Themen für Sie nachvollziehbarer zu machen, haben wir immer wieder Beispiele eingestreut, die sich auf die Entwicklung eines Smart-Home-Systems (SHS) beziehen.

Des Weiteren finden Sie Videos und animierte Grafiken zu ausgewählten Themen auf unserer Homepage. Der enstprechende Link ist in dem folgenden Icon angegeben:


Herzlichen Dank an alle SOPHISTen, die an der Broschüre mitgewirkt haben – das gesamte Requirements-Engineering und -Management-Buch-Team, die fachlichen Reviewer und die Diskussionspartner, die zu den Inhalten und der Qualität der Broschüre ihren Beitrag geleistet haben.

# 2. Was ist Requirements-Engineering?

Der Begriff Requirements-Engineering lässt sich frei übersetzen mit: Der Umgang mit Anforderungen nach den Prinzipien einer Ingenieurswissenschaft. Das Arbeiten mit Anforderungen sollte also wiederholbar, nachvollziehbar und begründet sein. Da sich Requirements-Engineering im Allgemeinen in ein Entwicklungsvorhaben (beispielsweise ein Projekt) mit gegebenen Randbedingungen wie Zeit und Geld einordnet, folgt noch eine weitere gewünschte Eigenschaft: Es sollte angemessen durchgeführt werden. Damit wollen wir sicherstellen, dass das Requirements-Engineering die Anforderungen in genau der Qualität erzeugt, die für die weitere Entwicklung benötigt werden.

Diese Gedanken spiegeln sich in der folgenden Definition wider:

**Definition der Disziplin Requirements-Engineering laut IREB [CPRE20]:**

Das Requirements-Engineering ist ein systematischer und disziplinierter Ansatz zur Spezifikation und zum Management von Anforderungen mit den Zielen

- die Wünsche und Bedürfnisse der Stakeholder zu verstehen und
- das Risiko zu minimieren, ein Produkt, das nicht diese Wünsche und Bedürfnisse erfüllt, auszuliefern.

Es ist nicht überraschend, dass in dieser Definition der Begriff der Anforderung eine zentrale Rolle spielt. Wir SOPHISTen haben eine für uns passende Definition einer Anforderung gefunden, die sich in der Praxis als gut verständlich, umfassend und ausreichend konkret erwiesen hat:

**Definition des Begriffs Anforderung nach SOPHIST:**

Eine Anforderung ist eine Aussage über eine Eigenschaft oder Leistung eines Produktes, eines Prozesses oder der am Prozess beteiligten Personen.

Damit fassen wir den Begriff der Anforderung breiter als üblich. Zum einen ist es mehr als das, was von dem zu entwickelnden System (dem Betrachtungsgegenstand der Anforderung) gefordert wird. Andererseits wollen wir uns hier nicht auf eine Repräsentationsform für Anforderungen festlegen. Für uns kann eine Anforderung auf unterschiedliche Weisen in einem Projekt dokumentiert oder vermittelt werden. Beispiele hierfür sind

- klassisch, textuell, ggf. nach Schablonen,
- als User-Story oder Epic,
- als Story im Rahmen eines Storytellings,
- modellbasiert.

Weiterhin wird in der Definition des Requirements-Engineerings der sogenannte Stakeholder in den Mittelpunkt gerückt. Er wird im Allgemeinen als eine Rolle definiert, die direkten oder indirekten Einfluss auf die Anforderungen hat. Ein direkter Einfluss bedeutet dabei, dass ein Stakeholder Anforderungen liefert (siehe Abschnitt 2.3 - „Quellen für Anforderungen“).

Die enge Zusammenarbeit zwischen dem Requirements-Engineer (auch Anforderungs- oder Systemanalytiker genannt) und den Stakeholdern erfordert einige Eigenschaften eines Requirements-Engineers, die über die methodische und fachliche Kompetenz in dem Anwendungsbereich des zu spezifizierenden Systems hinausgeht. Diese Eigenschaften haben sich in der Praxis als sehr wichtig herausgestellt, da sich ein großer Teil des Requirements-Engineerings mit der Kommunikation zwischen Personen mit verschiedenen Zielen, Vorbildungen und Eigenschaften beschäftigt.

In Abbildung 2.1 sind diese Eigenschaften nach IREB [CPRE20] dargestellt. Dort ist auch nach den grundlegenden (links) und den Kompetenzen unterschieden, die gerade für die Ermittlung der Anforderungen eine große Rolle spielen (rechts).

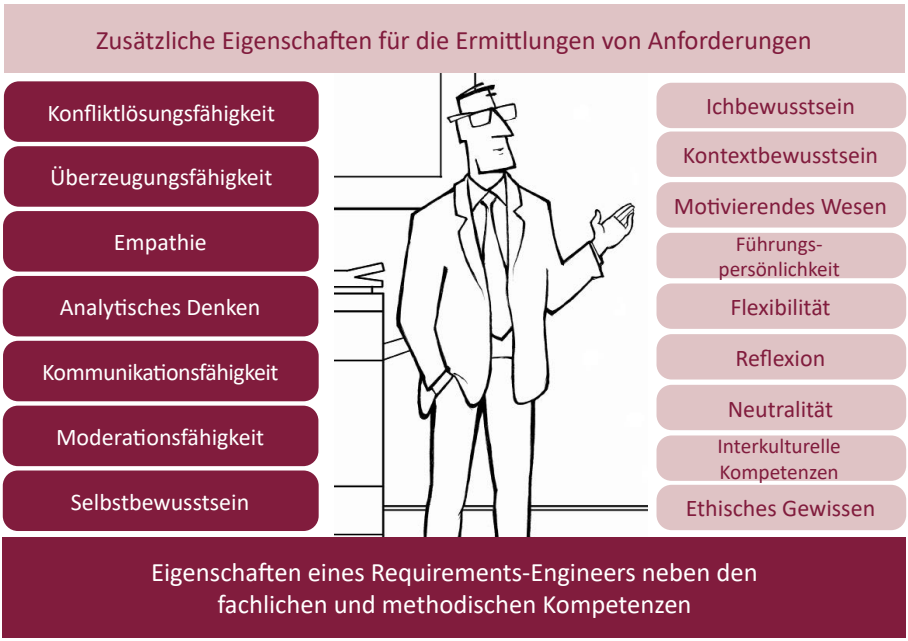



Abbildung 2.1: Eigenschaften eines Requirements-Engineers

In den folgenden Abschnitten werden wir nun einige Grundlagen vorstellen, die für das Arbeiten mit Anforderungen benötigt werden.

## 2.1 Klassifikation von Anforderungen

Prinzipiell haben Sie eine Vielzahl von Möglichkeiten, Ihre Anforderungen zu klassifizieren. Diese Einteilungen sollten immer einem bestimmten Zweck dienen. Wir stellen Ihnen hier die Arten von Klassifizierungen vor, die uns bei unserer Arbeit mit Anforderungen unterstützen und uns deswegen wichtig erscheinen. Jede Anforderung lässt sich bezüglich aller hier vorgestellten Klassifikationen einteilen, woraus Sie dann unter anderem die Aktionen im RE-Vorgehen ableiten können, die Sie für diese Anforderung durchführen sollten. So kann sich z.B. die Ermittlung von funktionalen von der Ermittlung von nicht-funktionalen Anforderungen unterscheiden, oder für verfeinerte Anforderungen sollte eine abstraktere Anforderung gefunden werden.

### Klassische Einteilung nach Typen

Die folgende Abbildung gibt Ihnen einen Überblick über die Anforderungstypen, die häufig in der Literatur unterschieden werden.


Abbildung 2.2: Arten von Anforderungen

Die beiden am häufigsten benötigten Typen sind:

- Funktionale Anforderungen: was stellt das System einer nutzenden Person oder einem Nachbarsystem an Funktionen unter gewissen Bedingungen zur Verfügung.
- Qualitätsanforderungen (Quality-of-Service-Requirements): sie geben die gewünschte Qualität des Systems, meist funktionsspezifisch, an (z. B. die Performance einer Funktion oder die Verfügbarkeit des gesamten Systems).

### Einteilung nach juristischer Verbindlichkeit

Die juristische Verbindlichkeit beschreibt den Grad der Bedeutung, den der Stakeholder den einzelnen Anforderungen beimisst. Man kann zwischen den folgenden Typen unterscheiden:

Für eine vollständige Anforderungssammlung sollten alle Verfeinerungsgrade von Anforderungen so spezifiziert werden, dass genügend Informationen für alle Beteiligten vorliegen. Das bedeutet jedoch nicht, dass jede Anforderung bis zum detailliertesten Verfeinerungsgrad verfeinert werden muss.

| Verbindlichkeit | Deutsches Schlüsselwort | Englisches Schlüsselwort |
|---|---|---|
| Pflicht | Muss | Shall |
| Wunsch | Sollte | Should |
| Absicht | Wird | Will |

Abbildung 2.3: Bewährte Schlüsselwörter für die rechtliche Verbindlichkeit

## Einteilung nach Verantwortlichkeit

Eine wichtige Unterscheidung zwischen Anforderungen ist der für eine Anforderung Verantwortliche. Wie wir weiter unten sehen werden, ist eine Aufgabe des Requirements-Engineerings, aus den Eingaben in ein Entwicklungsprojekt, den Ursprungsanforderungen, belastbare Systemanforderungen herzuleiten, womit eine Beziehung zwischen diesen beiden Typen von Anforderungen erstellt wird.

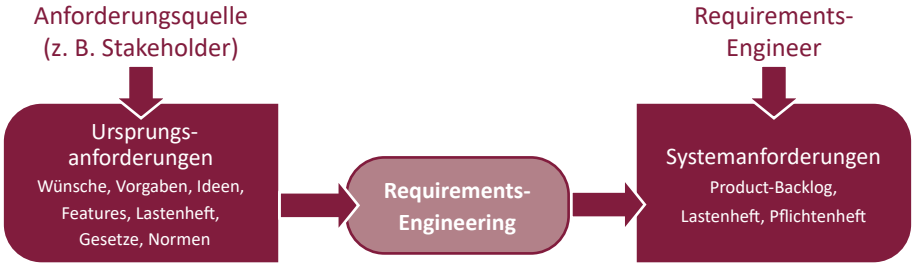



Abbildung 2.4: Ein- und Ausgaben des Requirements- Engineerings

## Einteilung nach dem Betrachtungsgegenstand

Die für das Requirements-Engineering wohl wichtigste Einteilung von Anforderungen wird durch ihren Betrachtungsgegenstand gegeben. Bezieht sich eine Anforderung auf das im Projekt betrachtete System, auf eine seiner Komponenten oder auf einen Geschäftsprozess, der durch das System unterstützt wird? Oder liegt die Anforderung vielleicht außerhalb des betrachteten Systems, in seinem Kontext? Gerade im Bereich des Systems-Engineerings, wo im Allgemeinen mehrere Ebenen des Systems betrachtet werden, sollten die Anforderungen eindeutig einem der relevanten Betrachtungsgegenstände zuordenbar sein (siehe Kapitel 9 - „Systems-Engineering“).

Wegen der Wichtigkeit dieser Charakteristik einer Anforderung wird diese Einteilung auch explizit in den Analysetätigkeiten beim Herleiten guter Anforderungen betrachtet (siehe Abschnitt 4.1 - „Tätigkeiten für die Analyse von Anforderungen“).

### Einteilung nach dem Verfeinerungsgrad

Die letzte Art der Einteilung erscheint zunächst nur in der Theorie relevant - spiegelt sich jedoch oft in Dokumentenlandschaften in der Praxis wider. Anforderungen können andere Anforderungen genauer beschreiben, geben also eine (fachliche) Lösung für eine Anforderung an. So kann ich zum Beispiel von einem Smart-Home-System fordern, dass es die Tür bei Annäherung einer bekannten Person die Haustür entriegeln soll. Genauere Aussagen, welche Prüfungen dabei durchgeführt werden sollen, würde man als Verfeinerung der vorigen Anforderung bezeichnen.

Diese Betrachtung liefert uns unter anderem eine klare Einordnung der Anforderungen und damit eine Möglichkeit eine gewisse Art von Vollständigkeit der Anforderungen nachvollziehbar zu gewährleisten.

## 2.2 Qualität von Anforderungen

Wie wir oben bereits erwähnt hatten, sollte das Requirements-Engineering angemessen durchgeführt werden. Dies kann unter anderem mit der Qualität der Anforderungen definiert werden, die in der Entwicklung erreicht werden sollte.

Auch für die Qualität von Anforderungen existieren zahlreiche unterschiedliche Ansätze. Allen gemein ist, dass sie diese Qualität über eine Menge von Qualitätskriterien definieren. Wir haben hier die Kriterien aufgelistet, nach denen wir unsere Anforderungen messen bzw. bewerten. Dabei unterscheiden wir zwischen Kriterien, die sich auf eine einzelne Anforderung beziehen und denen, die für eine Menge von Anforderungen gelten sollten.

Dabei unterscheiden wir zwischen Kriterien, die sich auf eine einzelne Anforderung beziehen und denen, die für eine Menge von Anforderungen gelten sollten.

### Anforderungen beim klassischen Vorgehen

In den Entwicklungsvorhaben, die ein klassisches Vorgehen verfolgen, müssen wir hohe Qualitätsansprüche an die Anforderungen stellen, da hier eine Verbesserung der einmal erzeugten Anforderungen nicht integraler Bestandteil des Vorgehens ist. Vielmehr geht man von der optimistischen Annahme aus, dass das Requirements-Engineering für den einmal erzeugten Teil der Anforderungen abgeschlossen ist.

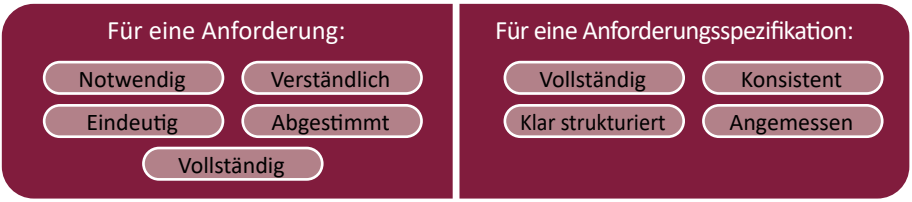



Abbildung 2.5: Qualitätskriterien für Anforderungen bei klassischen Vorgehen

### Anforderungen beim agilen Vorgehen

Für die Kriterien einer einzelnen Anforderung in agil durchgeführten Entwicklungsvorhaben (im Allgemeinen eine User-Story) verwenden wir das INVEST-Prinzip [Wake03].

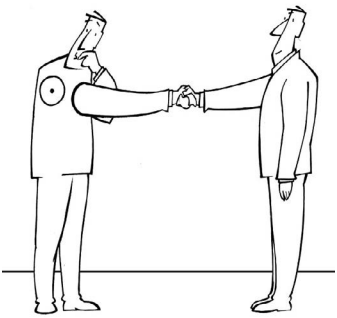

Abbildung 2.6: Das INVEST-Prinzip für eine User-Story

Die Kriterien für eine Sammlung von Anforderungen, also ein Backlog, unterscheiden sich kaum von denen, die für eine Anforderungsspezifikation im klassischen Umfeld gelten sollten.

## 2.3 Quellen für Anforderungen

Grundsätzlich gibt es drei verschiedene Quellen für Anforderungen:

- **Stakeholder:** Personen, Organisationen oder Institutionen, die direkt oder indirekt Einfluss auf das System haben.
- **Dokumente:** Gesetze, Normen, Handbücher oder sonstige Dokumentationen können zur Erhebung von Anforderungen genutzt werden.
- **Systeme:** Oft hilft es, ein Vorgängersystem oder ein Konkurrenzprodukt zu analysieren.



Am interessantesten ist hierbei die Gruppe der Stakeholder, da sie sehr häufig aus unterschiedlichen Bereichen kommen und somit unterschiedliche Ziele mit ihren Anforderungen an das zu entwickelnde System haben. Wir haben hier eine unvollständige Liste von Bereichen angegeben, die Anforderungen liefern oder sogar als Auslöser für ein Entwicklungsprojekt dienen können.

### Auftraggebende Organisation

In einer Beziehung zwischen auftraggebender und auftragnehmender Organisation ist die auftraggebende Organisation die wohl prominenteste Verursacherin eines Entwicklungsprojekts. Sie gibt der auftragnehmenden Organisation das Geld für die Entwicklung (und vielleicht auch für nachfolgende Tätigkeiten wie die Produktion) und bestimmt damit maßgeblich die Anforderungen an das zu entwickelnde Produkt.

### Innovations-/Portfolio-/Produktmanagement

Diese Bereiche hingegen sind oftmals interne Verursacher eines Entwicklungsprojekts. Sie sind dafür zuständig, die Produkte der Organisation weiter zu entwickeln, um am Markt konkurrenzfähig zu bleiben. Sie können in einem Projekt für einen Kunden jedoch auch zusätzliche Anforderungen fordern.

### Problem-/Änderungsmanagement

Das Problemmanagement liefert Änderungswünsche an ein System, die sehr häufig in späteren Lebenszyklusphasen (während der Produktion, des Transports, der Installation, des Betriebs) dieses Systems identifiziert wurden. Diese können zu zusätzlichen oder geänderten Anforderungen führen. Sie können auch den Bedarf für ein neues Entwicklungsprojekt auslösen, das dann die gewünschten Änderungen im Produkt umsetzt. Änderungen von Anforderungen in einer laufenden Entwicklung werden durch ein Änderungsmanagement unterstützt.

## 2.4 Haupttätigkeiten im Requirements-Engineering

Die Tätigkeiten, die ein Requirements-Engineer durchführen muss, lassen sich grob in vier Haupttätigkeiten einteilen. Im weiteren Verlauf dieser Broschüre werden wir auf diese Haupttätigkeiten im Einzelnen eingehen. In dieser Broschüre können wir nur die wichtigsten Aufgaben und Aspekte der einzelnen Haupttätigkeiten darstellen. Für eine ausführlichere Darstellung verweisen wir auf [Rupp20].

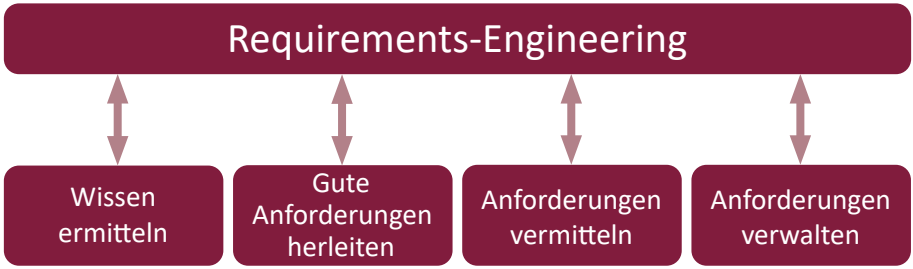



Abbildung 2.7: Die vier Haupttätigkeiten des Requirements-Engineerings

Auch wenn wir die Haupttätigkeiten in der oben angegebenen Reihenfolge vorstellen, so werden sich die einzelnen Tätigkeiten jedoch im Laufe der Systementwicklung oder -weiterentwicklung immer wiederholen. Sei es verursacht durch eine inkrementelle Betrachtung der Aufgaben oder dadurch, dass bei der Durchführung einer Tätigkeit die Notwendigkeit für andere Tätigkeiten aufkommt.

### Wissen ermitteln

Die erste Haupttätigkeit „Wissen ermitteln“ legt den Grundstein für das weitere Requirements-Engineering. Hier wird die Richtung für die Systementwicklung

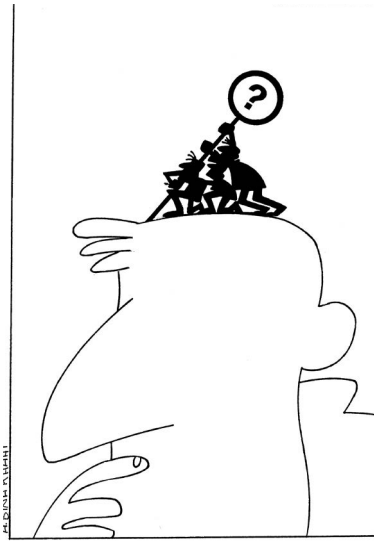
dadurch vorgegeben, dass die Visionen und die Ziele definiert werden, sofern sie nicht bereits vorliegen. Da nicht immer alle Anforderungsquellen, insbesondere die Stakeholder, bekannt sind, müssen diese als Grundlage für die weiteren Tätigkeiten ermittelt werden.

Wenn die Anforderungsquellen bekannt sind, können Sie damit beginnen, deren Anforderungen an das zu entwickelnde System zu ermitteln. Die Auswahl der jeweils für eine spezifische Entwicklung passenden Ermittlungstechniken ist dabei ein wichtiger Erfolgsfaktor.

## Gute Anforderungen herleiten

Um gute Anforderungen herzuleiten, sollten Sie zunächst die zuvor ermittelten Vorgaben analysieren, um ein möglichst umfassendes Bild der Anforderungen an das zu entwickelnde System zu bekommen. Je nach Vorgehensmodell muss dieses Bild nun auf den Prüfstand gestellt werden. VertreterInnen unterschiedlicher Rollen in Ihrer Entwicklung können die Anforderungen einem Review unterziehen:

- Aus Sicht der anforderungsgebenden Personen werden die Anforderungen bezüglich der Erfüllung der Erwartungen an das System überprüft.
- Für den Test des entwickelten Systems werden die Anforderungen daraufhin überprüft, ob aus ihnen entsprechende Testfälle abgeleitet werden können.
- Die Entwicklung (insbesondere die Architektur) wird die Realisierbarkeit der Anforderungen überprüfen.




In Abhängigkeit von Ihrer Anwendungsdomäne können noch weitere Überprüfungen notwendig werden. So kann im Automotive-Umfeld die Überprüfung auf Erfüllung von Vorgaben bzgl. der funktionalen Sicherheit ([ISO26262]) gefordert sein.

## Anforderungen vermitteln

Mit den bisher betrachteten Haupttätigkeiten sind Sie in der Lage, sich selbst einen guten Überblick über die von Ihnen benötigten Anforderungen zu verschaffen. Da aber die Anforderungen jemand anderem bei seiner Arbeit helfen sollen, müssen diese den beteiligten Empfängern vermittelt werden.Hierbei gehen wir davon aus, dass Sie entweder die Anforderungen dokumentieren wollen oder Ihr Wissen über die Anforderungen in anderer Art und Weise (Storytelling, Videos) weitergeben wollen.

Falls Sie eine Sammlung von dokumentierten Anforderungen erstellen müssen oder wollen, haben Sie die Wahl zwischen zwei Alternativen:

- Sie können die Anforderungen an das System natürlichsprachlich formulieren.
- Sie können sich entscheiden, die Anforderungen modellbasiert zu dokumentieren.

In unseren Beratungstätigkeiten hatte immer ein gemischter Ansatz aus beiden Alternativen den höchsten Mehrwert und die beste Akzeptanz.

## Anforderungen verwalten

Sobald Sie die Dokumentation von Anforderungen betrachten, kommt eine neue Herausforderung auf Sie zu: das Verwalten dieser dokumentierten Anforderungen. Hierfür müssen Sie einige Entscheidungen treffen:


- Aus welchen Informationsarten setzen sich die benötigten Anforderungssammlungen zusammen und wie sind diese im Einzelnen aufgebaut?
- Wer darf welche Aktionen in den Anforderungen durchführen (Vergabe von Rechten und Rollen)?
- Welche zusätzlichen Informationen benötigen Sie zum Verwalten der Anforderungen? Zum Beispiel: aktueller Zustand der Anforderung, Varianteninformationen.
- Wie soll die Nachverfolgbarkeit (Traceability) sichergestellt werden.
- Welches Versionierungskonzept möchten Sie Ihren Anforderungen zugrunde legen?
- Welche Informationen müssen in Ihrer Systementwicklung in welcher Form bereitgestellt werden?

Diese und viele weitere Entscheidungen und Tätigkeiten ordnen wir dem Verwalten der Anforderungen zu, um die benötigten Informationen zu jeder Zeit nachvollziehbar zur Verfügung stellen zu können.

# 3. Wissen ermitteln

Die Wissensermittlung ist nicht nur eine der ersten, sondern auch eine der wichtigsten Tätigkeiten des Requirements-Engineerings. Damit sind die Anforderungen selbst gemeint, aber auch weitere Informationen, wie beispielsweise Quellen, Abgrenzung und Geschäftsprozesse. Sie findet in jedem denkbaren Szenario, jedem Vorgehensmodell und für jeden Verfeinerungsgrad von Anforderungen statt und gestaltet sich im Detail jedes Mal anders. Sollten Sie es nicht schaffen, die richtigen Stakeholder zu finden, den Kontext richtig zu setzen oder aufgrund der Geschäftsprozesse das richtige Wissen zu ermitteln, dann ist Ihr Requirements-Engineering sehr wahrscheinlich zum Scheitern verurteilt. Das heißt natürlich nicht, dass all diese Tätigkeiten gleich am Anfang finalisiert werden müssen – vielmehr ist die Ermittlung ein Prozess, der Sie während des gesamten Requirements-Engineerings begleitet.

Im Bereich der Anforderungsermittlung hat sich in den letzten Jahren sehr viel verändert. Die Digitalisierung und Konzepte wie Smart Cities oder Smart Rural Areas haben eine signifikante Auswirkung auf unseren sozialen Kontext und unser menschliches Miteinander. IT hält dadurch in vielen Bereichen Einzug, die bisher kaum von ihr betroffen waren. Natürlich haben auch jetzt schon die meisten Menschen eine Waschmaschine im Haus stehen, die Software beinhaltet, doch hier wurden die Wünsche der BenutzerInnen häufig lediglich erahnt. Sobald die Digitalisierung merklich in Bereiche wie unser Gesundheitssystem, die Warenzustellung, das Transportsystem und die Kommunikation eingreift, ist es wichtig, die Bedarfe der unterschiedlichsten Stakeholder im Detail zu verstehen. Wir adaptieren die RE-Methoden in unseren Beratungen immer sehr in Bezug auf den Kontext, in dem sie angewendet werden, und von den dort beheimateten Stakeholdern. Wenn die Digitalisierung in alle Lebensbereiche einzieht, dann sind von unseren Systemen auch Menschen betroffen, die bisher wenig mit IT zu tun hatten und denen ihre Rolle als Stakeholder alles andere als vertraut ist.

## 3.1 Ziele, Quellen und Systemkontext

Doch fangen wir am Anfang an. Jedes Vorhaben wird aus einem bestimmten Grund oder zu einem bestimmten Zweck unternommen, benötigt Quellen für Anforderungen und sollte einen abgesteckten Rahmen haben, um Notwendiges von Unnötigem zu unterscheiden.

### Ziele und Zielfindung

Das Festlegen der Ziele für das System und ein Projekt hat einen großen Einfluss auf den Erfolg der Entwicklung. Jede Systementwicklung sollte mit der Definition von

Zielen im Umfang von ca. einer halben Seite starten. Wir haben in der Praxis gute Erfahrungen damit gemacht, dies in Form einer Zieletabelle oder auch mittels eines Produkt-/Projekt-Canvas zu notieren. Wenn Ziele nicht dokumentiert oder nur unklar definiert sind, existiert keine Ausgangsbasis für die nachfolgenden Schritte, Tätigkeiten oder Aktivitäten. Sie als Requirements- Engineer haben dann keine Vorgabe, welche Ziele durch die Anforderungen erfüllt werden müssen, und spezifizieren möglicherweise an den eigentlichen Zielen vorbei.

Der Prozess der Zielfindung hängt stark von den gegebenen Rahmenbedingungen ab. Erfinden Sie ein neues Produkt, dann sind vor allem visionäres Denken und Offenheit bezüglich aller erdenklichen Lösungen gefragt. Haben Sie vor ein Altsystem abzulösen, dann sollten Sie sich vor allem um mögliche Optimierungen kümmern und müssen deren Auswirkungen auf bestehende Funktionen betrachten.

### Anforderungsquellen

Die Suche nach den relevanten Anforderungsquellen ist entscheidend für Ihren Erfolg. Typische Anforderungsquellen sind Dokumente, Systeme und natürlich Stakeholder. Halten Sie als Requirements-Engineer die Augen aber auch darüber hinaus nach allen erdenklichen Informationsquellen für Anforderungen offen, bzw. schaffen Sie sich Hilfsmittel, falls benötigte Quellen nicht nutzbar sind. Ein Lösungsansatz besteht z. B. darin, mit Personas [Goodwin09] als fiktive Repräsentanten für real existierende Menschen zu arbeiten.

### Systemumfang und -kontext

Als letzte Tätigkeit, bevor Sie in die eigentliche Ermittlung von Anforderungen einsteigen können, gilt es festzulegen, was zum System gehört (Scope) und was sich außerhalb des Systems befindet (Kontext) und damit mit dem System interagieren kann.

Wird der Systemumfang oder -kontext im Rahmen des Requirements-Engineerings falsch oder unvollständig berücksichtigt, führt dies zu unvollständigen oder fehlerhaften, ggf. sogar zu nicht notwendigen, also zu vielen Anforderungen.

## 3.2 Anforderungsermittlung

Ziel der Anforderungsermittlung ist es, mit möglichst geringem Aufwand und angepasst an die Rahmenbedingungen des Vorhabens, die Anforderungen zu erfassen, die die Entwicklung eines Systems erlauben, das den Stakeholdern möglichst viel Gewinn bringt. Wir suchen deshalb gerade für die Anforderungsermittlung einerseits nach dem effizientesten Mittelweg zwischen Risikoreduktion und Kostenexplosion und andererseits nach professionellen Mitteln, die Anforderungen aus den jeweiligen Quellen zu erheben. Das ermittelte Wissen ist das Rohmaterial, aus dem dann gute Anforderungen hergeleitet, dokumentiert, vermittelt und verwaltet werden.

### Vorbedingungen für eine gute Ermittlung

Rechnen Sie nicht damit, dass Ihnen die Stakeholder perfekte Anforderungen auf

dem Silbertablett präsentieren. Anforderungsermittlung ist harte Arbeit. Aus diesem Grund sollten Sie die wichtigsten Faktoren für eine erfolgreiche Anforderungsermittlung kennen:

- Wissen über die grundlegende Funktionsweise zwischenmenschlicher Kommunikation
- Wissen über Repräsentationssysteme der Sprache
- die Kenntnis der Stärken und Schwächen der einzelnen Ermittlungstechniken
- Analyse der relevanten Rahmenbedingungen für den Einsatz von Ermittlungstechniken
- Wissen über Auswahl und Kombination von geeigneten Ermittlungstechniken
- die richtige Atmosphäre für den Einsatz von Ermittlungstechniken schaffen (insbesondere für Kreativitätstechniken)

## Kriterien für die Auswahl von Ermittlungstechniken

Jede Ermittlungstechnik hat Stärken und Schwächen und eignet sich dadurch nur für den Einsatz unter den passenden Rahmenbedingungen. Diese Rahmenbedingungen zu kennen und sie für die Auswahl einer geeigneten Ermittlungstechnik zu nutzen, entscheidet maßgeblich über den Erfolg der Anforderungsermittlung.

- Wie groß ist das Wissen des Stakeholders in Bezug auf den Betrachtungsgegenstand?
- Wie hoch ist die Motivation der Stakeholder an der Ermittlung mitzuwirken?
- Ist der Stakeholder jemand, der gerne selbst aktiv wird? Redet er gerne oder ist er eher haptisch veranlagt?
- Wie ist die örtliche Verteilung der Stakeholder?
- Wie ist die zeitliche Verfügbarkeit der Stakeholder?
- Gibt es besondere Gruppendynamik zwischen den Stakeholder zu beachten?

Dies ist nur eine kleine Auswahl von Kriterien, lässt aber erahnen, dass auch die Auswahl der geeigneten Ermittlungstechniken von der Erfahrung des Requirements-Engineers abhängt.

## Vier Gruppen von klassischen Ermittlungstechniken

Um Wissen zu ermitteln, wurde eine Vielzahl von Techniken entwickelt, welche sich grob in vier Gruppen gliedern lassen.

- **Befragungstechniken**
  (Fragebogen und Interview) sind die Klassiker unter den Ermittlungstechniken und basieren darauf, die Stakeholder gezielt nach ihren Wünschen und Bedürfnissen zu befragen, um aus den Antworten Anforderungen ableiten zu können.

- **Beobachtungstechniken**
  (Feldbeobachtung, Apprenticing, Contextual Inquiry) werden eingesetzt, wenn

die Stakeholder ihr Wissen nicht sprachlich ausdrücken können oder viele implizite Wünsche erwartet werden.

- **Artefaktbasierte Techniken**
  (Systemarchäologie und Wiederverwendung) haben ihre Stärken, wenn kein Wissensträger mehr verfügbar ist und daher die Fachlogik nur aus dem System selbst und seiner Dokumentation ermittelt werden kann.

- **Kreativitätstechniken**
  (Brainstorming oder Brainstorming Paradox) werden eingesetzt, wenn innovative Ideen gefordert sind.

### Neue Ansätze/Frameworks – Co-Creation-Modelle, CrowdRE und Living Labs


Neben den eben erläuterten Ermittlungstechniken gibt es eine ganze Menge an Frameworks, die mehrere Ermittlungstechniken kombinieren.

Als Vertreter möchten wir das CrowdRE herausgreifen. Hier wird versucht, eine bisher anonyme Menge an Menschen (Crowd) über einen möglichst niedrigschwelligen Zugang zur Mitarbeit zu motivieren. Dabei gibt es mannigfaltige Ansätze, wie die Crowd gewonnen wird und mit welchen Mitteln sie ihren Beitrag leisten kann. Meist sind hier elektronische Wege der Standardfall.

Ziel von CrowdRE ist es auch, sogenannte Unicorns/Einhörner (hoch motivierte Stakeholder mit viel Wissen) in der Menge an Antwortenden zu finden. Diese versucht man dann zu einer weitergehenden Mitarbeit zu animieren.

## 3.3 Sprachliche Effekte aufspüren – Das SOPHIST-*RE*gelwerk


Beim Arbeiten mit Anforderungen, insbesondere beim Ermitteln, kann es zu unerwünschten sprachlichen Effekten kommen. Sowohl bei einer direkten Kommunikation als auch beim Schreiben oder Lesen von geschriebenen Anforderungen. Der Einsatz des SOPHIST-*RE*gelwerks hilft dabei, die unerwünschten Effekte zu reduzieren und damit die Basis für qualitativ hochwertige Anforderungen zu schaffen. Doch was sind sprachliche Effekte überhaupt und wodurch entstehen sie?

### Sprachliche Effekte

Jeder Mensch nimmt seine Umwelt anders wahr. Die Gesamtheit seiner Wahrnehmungen bildet das persönliche Wissen des Einzelnen. Dies wird beeinflusst durch Vorwissen, die soziale Prägung sowie die gesammelten Erfahrungen.

Sobald Menschen ihr Wissen kommunizieren, finden Transformationsprozesse statt, die davon abhängig sind, wie die Kommunizierenden die Situation und ihr Gegenüber einschätzen, welches Vorwissen beim Anderen vorausgesetzt wird oder wie sicher man sich eines Sachverhalts ist. Diese Transformationsprozesse können in einem Verlust oder einer Verfälschung von Informationen resultieren.

Transformationen lassen sich aber aufspüren und auflösen – Voraussetzung dafür ist, dass der Requirements-Engineer die möglichen Transformationsarten und deren Konsequenzen kennt. An dieser Stelle setzt das SOPHIST-*RE*gelwerk an, dass im Wesentlichen auf dem Metamodell der Sprache und der Neurolinguistischen Programmierung (NLP) beruht [Bandler75] [Bandler94]. Bandler und Grinder unterscheiden zwischen drei „Arten der Umgestaltung“: Tilgung, Generalisierung und Verzerrung.

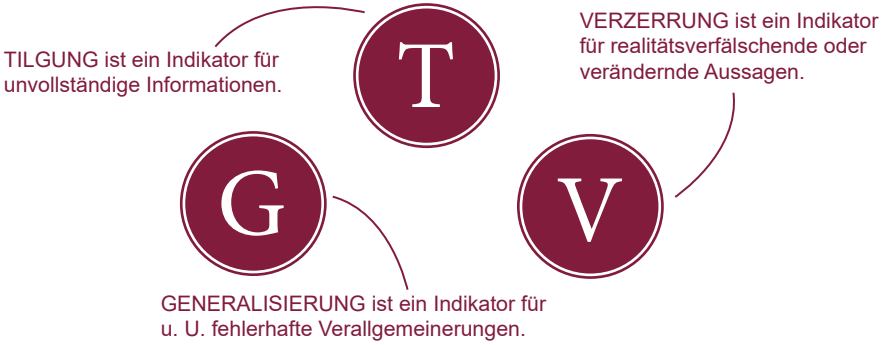



Abbildung 3.1: Klassifizierung sprachlicher Effekte

## Das SOPHIST-*RE*gelwerk

Glücklicherweise geht der Mensch beim natürlichsprachlichen Formulieren von Wissen regelgeleitet vor. Somit gibt es in natürlichsprachlichen Aussagen, gleich ob in Wort oder Schrift, Anzeichen dafür, wenn einer oder mehrere der oben genannten Transformationsprozesse stattgefunden haben. Das SOPHIST-*RE*gelwerk beruht auf diesen unbewusst angewendeten Regeln und ermöglicht Ihnen, in der Systemanalyse auf eine definierte und systematische Art und Weise mehrdeutige und widersprüchliche Aussagen in Anforderungsdokumenten zu finden.


### Prozessdetails ergänzen

Prüfen Sie, ob bereits alle relevanten Prozessdetails in der Anforderung ausreichend benannt sind. Hinterfragen Sie das Verb (oder ein von einem Verb abgeleitetes Eigenschaftswort), das den Prozess ausdrückt mit den W-Fragewörtern: Wann? Wem? Wo? Wie oft? etc. Entscheiden Sie, ob die fehlenden Informationen für die Umsetzung essenziell sind und ergänzen Sie gegebenenfalls die noch fehlenden Prozessdetails.

Wir haben hier nur eine von insgesamt 17 Regeln angegeben, um Ihnen einen Eindruck von diesen Regeln zu geben.

Aber keine Angst, Sie müssen zu Beginn nicht alle 17 Regeln auswendig lernen. Suchen Sie sich die zwei bis drei für Sie wichtigsten Regeln raus, und horchen Sie auf die entsprechenden Signalwörter. Und stellen Sie bei Bedarf dann die notwendigen Fragen, um z.B. die getilgten Informationen zu erhalten.

Weitere Informationen zu der Anwendung des *RE*gelwerks finden Sie auch in dem folgenden Video:


| | |
|---|---|
| **Signalwörter** | Verben: speichern, anzeigen, löschen, berechnen, montieren …<br><br>Von einem Verb abgeleitetes Adjektiv: gespeicherte Datei, freigegebenes Dokument, konfigurierter Zeitplan …<br><br>Von einem Verb abgeleitetes Adverb: komprimiert übertragen, systemgesteuert drucken … |
| **Beschreibung** | Um in einer Anforderung einen Prozess eindeutig zu beschreiben, ist es notwendig, dass alle zur vollständigen Erklärung notwendigen Informationen vorhanden sind. Dies betrifft Verben oder aus einem Verb abgeleitete Adjektive oder Adverbien. Bleiben zum Prozess noch Fragen offen, so müssen die zugehörigen Informationen aufgedeckt und der Anforderung hinzugefügt werden.<br><br>Stellen Sie konkrete Fragen wie „Durch wen oder was wird der Prozess ausgeführt?“, „Wie oft wird der Prozess ausgeführt?“, „Wie führen Sie als Benutzer den Prozess fachlich aus?“, „Wann bzw. unter welchen Randbedingungen wird der Prozess ausgeführt?“.<br><br>Die Ergänzung der unvollständig spezifizierten Prozessdetails kann zu aufwendigen Umformulierungen der Anforderung führen. Häufig resultieren aus den Antworten zu den Fragen aber auch zusätzliche Anforderungen oder man erkennt, dass die Originalanforderung durch mehrere detailliertere Anforderungen verfeinert und somit ersetzt werden muss.<br><br>Denke Sie auch an Aspekte, die durch Eigenschaften beschrieben sind. Sie treten entweder als Adjektive oder Adverbien auf. In diesem Fall stellen Sie Fragen, die spezifische Informationen zur Eigenschaft bestimmen. |
| **Beispiel** | Ursprungsanforderung: *„Falls die Identitätsprüfung nicht korrekt ist, muss das Smart-Home-System dies **anzeigen**.“*<br><br>Noch unklar: Was wird angezeigt? Wem wird angezeigt? Wann wird angezeigt? Wie lange wird angezeigt? Etc.<br><br>Verbesserte Ergebnisanforderung:<br><br>***Nachdem das Smart-Home-System die vom Benutzer eingegebene Identität geprüft hat** und **falls die Identitätsprüfung nicht korrekt ist,** muss das Smart-Home-System **dem Benutzer** die **Fehlermeldung „Zutritt wurde verweigert“ für drei Sekunden** anzeigen**.*** |
| **Tipps & Tricks** | Prozessdetails sind Angaben/Informationen „rund um“ ein Verb/Adjektiv/Adverb, die mittels der W-Fragen analysiert werden können, aber nicht zwangsweise formuliert werden müssen. Denken Sie daran, dass nicht immer alle Prozessdetails in jede Anforderung aufgenommen werden müssen. Manche Prozessdetails sind bereits durch eine Vorbedingung des Use-Cases gesetzt oder sind mit einer vorherigen Anforderung bekannt und müssen dann nicht erneut formuliert werden. |

# 4. Gute Anforderungen herleiten

Nachdem die Anforderungen Ihrer Stakeholder (im Folgenden Ursprungsanforderungen genannt) ermittelt wurden, müssen aus diesen nun Anforderungen hergeleitet werden, die gut genug sind, um als Basis für die Entwicklung und Test zu dienen.

Der Erfahrung nach besteht ein häufiges Manko in den Ursprungsanforderungen darin, dass sie mehr als das fordern, was Ihr Entwicklungsgegenstand, Ihr System, zu leisten vermag. Deswegen sollten Sie bei der Analyse der Ursprungsanforderungen besonderes Augenmerk darauf legen, was Sie daraus für Ihr System herleiten. Diese so hergeleiteten Anforderungen bezeichnen wir im Folgenden als Systemanforderungen.

Nachdem Sie Ihre Systemanforderungen identifiziert haben, können Unterschiede zu den Ursprungsanforderungen bestehen, weil Sie

- Freiheiten in der Interpretation in den Ursprungsanforderungen ausgenutzt haben,
- weitergehende Festlegungen getroffen haben,
- Anpassungen in den Anforderungen vornehmen mussten,
- fehlende Anforderungen ergänzen mussten.

Diese (und noch viele andere) Gründe führen zu der Notwendigkeit, die gefundenen Systemanforderungen von den Stakeholdern prüfen zu lassen und bei Bedarf Unstimmigkeiten zu beheben.

## 4.1 Tätigkeiten für die Analyse von Anforderungen

Mithilfe der hier vorgestellten Tätigkeiten werden Sie viele, eventuell bislang fehlende Systemanforderungen erzeugen. Dabei gehen wir davon aus, dass jede Anforderung eine andere Anforderung verfeinert. Eine Ausnahme bildet dabei die abstrakteste Ebene von Anforderungen. Diese Anforderungen stehen nebeneinander und bilden die Basis für die verfeinernden Anforderungen. Somit ergeben sich viele Bäume (mathematisch als „Wald“ bezeichnet), wobei die Wurzeln durch die abstraktesten Anforderungen gebildet werden und die Blätter durch die Anforderungen repräsentiert werden, die nicht mehr verfeinert werden.

Bei einem Use-Case-basierten Ansatz repräsentieren die Wurzeln der Anforderungsbäume dabei entweder

- die Use-Cases des Systems oder

- die Kategorien der nicht-funktionalen Anforderungen, die nicht den funktionalen Anforderungen zugeordnet werden können.

Ein unvollständiges Beispiel dazu ist in der folgenden Abbildung gegeben.

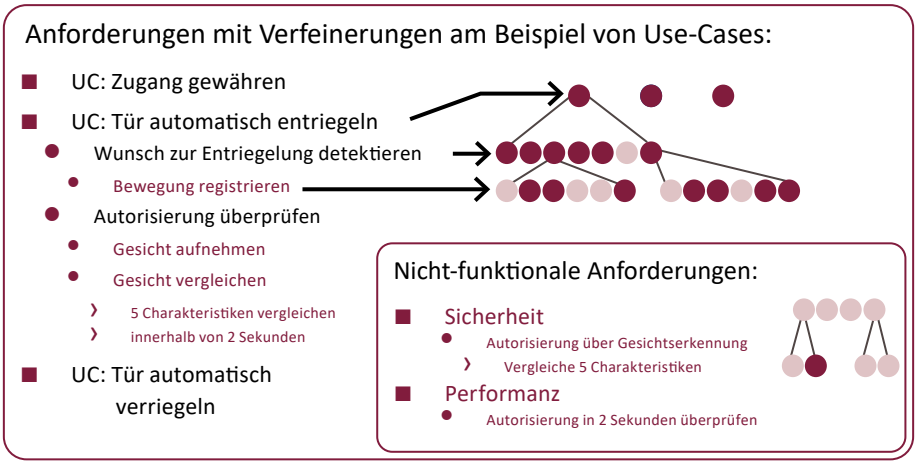



Abbildung 4.1: Zusammenhang zwischen Anforderungen

Bitte beachten Sie, dass jede der im Folgenden kurz vorgestellten Tätigkeiten aus gegeben Anforderungen (Ursprungs- oder bereits erzeugte Systemanforderungen) neue Systemanforderungen erzeugt oder die gegebenen Anforderungen verändert.

In dem ersten Schritt **Anforderungen separieren** sollten Sie, falls notwendig, eine gegebene Ursprungsanforderung in mehrere Anforderungen zerlegen, um diese in den nächsten Schritten getrennt voneinander betrachten zu können.

Mit dem zweiten Schritt **notwendige Anforderungen extrahieren** können Sie überprüfen, ob sich die separierten Anforderungen auch wirklich an das betrachtete System richten. Ist dies nicht der Fall, so müssen Sie den für Ihr System relevanten Anteil für die weitere Betrachtung identifizieren. Diese beiden ersten Schritte sollten Sie auf alle Ursprungsanforderungen anwenden, um so einen guten Startpunkt für die weitere Analyse zu erzeugen und um sicher zu sein, alle Ursprungsanforderungen betrachtet zu haben.

Die nächste Aufgabe ist es,die gegebenen **Anforderungen abstrahieren**, bis Sie die oberste Ebene Ihrer Anforderungen, also die Wurzeln der zu erzeugenden Bäume, identifiziert haben. Dabei können Sie sowohl auf neue Anforderungen stoßen als auch bereits bestehende Anforderungen in eine Abstraktions-/Verfeinerungshierarchie einordnen. Die oberste Ebene wird dann in dem Schritt **fehlende Anforderungen ergänzen** vervollständigt, in dem Sie weitere Anforderungen auf dieser Ebene definieren können.

Damit haben Sie den Startpunkt für die vorletzte Tätigkeit gefunden, in der Sie **Anforderungen verfeinern** und dazu entscheiden können welche der bestehenden Knoten in den Bäumen weiter verfeinert werden sollen.

Zum Abschluss der Analyse können Sie die gefundenen **Anforderungen verbessern** und sie dabei ihre individuelle Qualität überprüfen, um sie z. B. eindeutig zu formulieren.

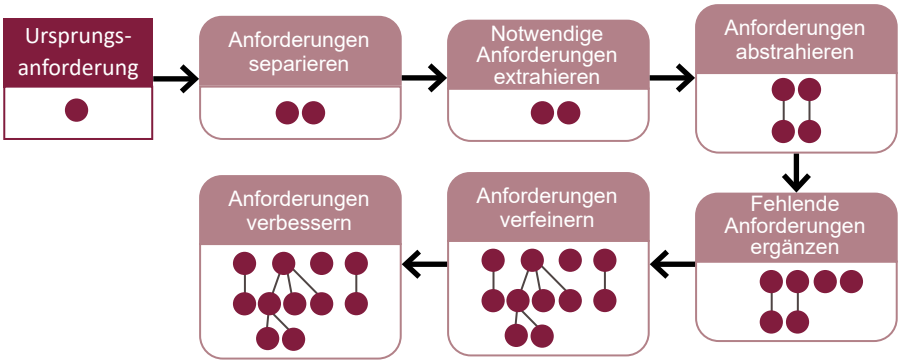



Abbildung 4.2: Die Analyseaufgaben in der Übersicht

GIF www.sophist.de/bre/k4a2/

Die aus den Tätigkeiten entstandene Gliederungsstruktur kann die Basis Ihrer Anforderungssammlung der natürlichsprachlichen sowie der modellbasierten Anforderungen bilden.

Wir unterstützen die hier vorgestellten Tätigkeiten durch verschiedene Techniken. So können Sie unterschiedliche Fragen stellen, die Sie zu den Systemanforderungen hinleiten. Zum Beispiel hilft Ihnen die Frage nach dem WOZU einer Anforderung bei dem Schritt des Abstrahierens. Die formale Betrachtung von gegebenen Schnittstellen kann Ihnen fehlende Anforderungen liefern. Allgemein werden viele Tätigkeiten durch die Regeln des SOPHIST-*RE*gelwerks (siehe Abschnitt 3.3 - „Sprachliche Effekte aufspüren – Das SOPHIST-*RE*gelwerk“) unterstützt.

In der Praxis werden Sie bei der Durchführung der einzelnen Tätigkeiten Wissen benötigen, das über die Informationen in den Ursprungsanforderungen hinausgeht. Um diese Wissenslücken auszugleichen, werden Sie Ihre Stakeholder bzw. Auftraggebenden fragen müssen oder Annahmen treffen, die im Nachhinein dann noch abgesichert werden müssen.

Die beispielhafte Anwendung der Analysetätigkeiten haben wir in dem folgenden Video dargestellt:

www.sophist.de/bre/k4v1

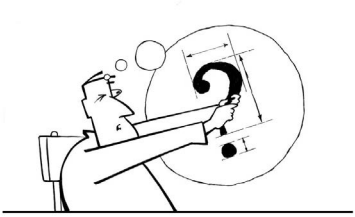
Bitte beachten Sie, dass Sie diese Tätigkeiten in Ihren RE-Prozess einbetten müssen. Sie müssen zum Beispiel festlegen, wann Sie die Analyse für welchen Teil der

Ursprungsanforderungen durchführen. Weiterhin ist es wichtig, Abbruchkriterien für die einzelnen Tätigkeiten zu definieren, da Sie sonst vielleicht mehr Aufwand als benötigt in die Analyse der Ursprungsanforderungen investieren.

## 4.2 Anforderungen prüfen und abstimmen

Nachdem Sie Ihre Anforderungen aus den Ursprungsanforderungen abgeleitet haben, ist es an der Zeit, durch Prüfen und Abstimmen dieser Anforderungen sicherzustellen, dass sie den festgelegten Qualitätskriterien genügen und jeder Stakeholder mit den Anforderungen zufrieden ist.

### Anforderungen prüfen



Fehlerhafte Anforderungen beeinträchtigen die Entwicklungsaktivitäten. Je später ein Fehler festgestellt wird, desto mehr Änderungen müssen nachgepflegt werden – sei es in Architekturbeschreibungen, in Testartefakten oder sogar im Quellcode. Für eine Prüfung von Anforderungen empfiehlt es sich, zunächst die Ziele der Prüfung festzulegen. Von diesen hängt dann die Auswahl der jeweiligen Prüftechniken ab. Nach der Vorbereitung und Durchführung der Prüfung können Sie dann die Ergebnisse einarbeiten. Je nach Umfang der Änderungen bietet sich dann eine eventuelle erneute Prüfung an.


Bei den Prüfungen können Sie prinzipiell zwischen immer wieder durchgeführten, automatisierten Prüfungen und Prüfungen zu bestimmten Meilensteinen in Ihrer Entwicklung unterscheiden.

### Anforderungen abstimmen

Im Rahmen des Requirements-Engineerings, insbesondere bei der Prüfung der Anforderungen, können an vielen Stellen Unstimmigkeiten auftreten. Diese reichen von fachlichen Missverständnissen bis hin zu schweren persönlichen Konflikten, die ohne zusätzliche Hilfe nicht behoben werden können.

Die Konflikte, die wir meistens im Rahmen des RE-Prozesses behandeln müssen, beschreiben Unvereinbarkeiten von Anforderungen, die auf widersprüchlichen Wahrnehmungen oder unterschiedlichen Zielsetzungen der Stakeholder basieren.

Die Aufgabe des Requirements-Engineers ist es, diese Konflikte zu identifizieren, sie zu analysieren und festzustellen, wie der Konflikt gemeinsam mit den beteiligten

Stakeholdern gelöst werden kann. Im Anschluss an eine Konfliktlösung empfiehlt es sich, den Konflikt sowie den Prozess und das Ergebnis der Lösungsfindung zu dokumentieren, um eine Lösung für ähnliche Konflikte mit weniger Aufwand erarbeiten zu können.


Für jeden dieser Schritte existiert eine Anzahl von Techniken, aus denen Sie aufgrund Ihrer Projektrandbedingungen (z. B. Verfügbarkeit der Stakeholder zur Lösung des Konflikts) auswählen können. Für eine Auflösung von Konflikten existieren Techniken von einer Kompromissbildung bis hin zu der Ober-sticht-Unter-Technik. Gerade bei dem Schritt der Identifikation von Konflikten zwischen Anforderungen hilft Ihnen eine strukturierte und nachvollziehbare Dokumentation der Anforderungen.

# 5. Anforderungen dokumentieren und vermitteln

Egal ob mit oder ohne Spezifikation, modellbasiert oder natürlichsprachlich – ob als Erzählung, Backlog oder als Spezifikation - um die Anforderungen möglichst gut an andere vermitteln zu können, muss die Vermittlung geplant sein und es müssen verschiedene Einflussfaktoren zur Auswahl der geeigneten Technik berücksichtigt werden. Schließlich ermitteln und analysieren wir die Anforderungen meist nicht für uns selbst, sondern für andere am Entwicklungsprozess beteiligte Rollen wie Entwicklung, Test, Systemarchitektur und viele mehr. Demnach ist eine wichtige Aufgabe für Sie als Requirements-Engineer, die Anforderungen an andere Personen zu vermitteln, und zwar so, dass diese die Anforderungen auch verstehen und keine Missverständnisse und Fehlinterpretationen auftreten. Machen Sie sich Gedanken, wie die Vermittlung stattfinden soll. Sie können zum Beispiel alle Anforderungen aufschreiben (dokumentieren) und den Empfängern zum Lesen geben. Oder Sie unterhalten sich mit den Personen über die Anforderungen bis hin zum gemeinsamen spielerischen Erleben der Anforderungen. Unserer Projekterfahrung zeigt, dass eine erfolgreiche Vermittlung meist auf einen durchdachten Mix aus verschiedenen Techniken besteht.

## 5.1 Anforderungen ohne Dokumentation vermitteln

Anforderungen und User-Storys lassen sich auch ohne eine klassische Anforderungsdokumentation vermitteln. Gerade in der agilen Welt haben wir sehr positive Erfahrungen mit Techniken, bei denen wenig dokumentiert und viel kommuniziert wird (z. B. User-Storys oder Storytelling). Allerdings schließen sich Anforderungsdokumentation mit Modellen oder natürlichsprachliche Anforderungsdokumentationen und die im Folgenden vorgestellten Möglichkeiten nicht gegenseitig aus. Vielmehr können Sie diese durchaus kombinieren, um die Stärken beider Varianten zu nutzen.

### Storytelling

Hören Sie sich gerne einen nüchternen Sachvortrag an, wenn Sie den gleichen Inhalt auch in einer guten Story verpackt genießen könnten? Geschichten sind etwas, was Menschen seit eh und je fasziniert. Mittels Geschichten wird seit Anbeginn der Menschheit Wissen vermittelt. Somit ist es nicht verwunderlich, dass das Erzählen von Geschichten (Storytelling) auch im Requirements-Engineering zur Wissensvermittlung eingesetzt wird. Wir nutzen dabei unterschiedliche Arten von Storys.

- **Hintergrundstorys** erzählen etwas über die Umgebung und die Nutzung des Systems – sie übermitteln wichtige Hintergrundinformationen.
- **Personality-Storys** stellen eine Persona vor und machen sie erlebbar.
- **Überzeugungsstorys** transportieren ganz am Anfang des Requirements-Engineerings die Ausgangslage und motivieren, warum das System gebraucht wird.
- **Erklärungsstorys** verdeutlichen einzelne Abläufe und Verhaltensweisen des Systems. Sie kommen unserer Erfahrung nach im RE am häufigsten zum Einsatz.

## User-Story und Story Mapping

User-Storys beschreiben gewünschte Funktionalitäten bzw. Eigenschaften eines Systems aus der Sicht derjenigen Person, welche die Funktionalität bzw. Eigenschaft benötigt. In der agilen Welt ist die Vermittlung von Anforderungen mittels User-Storys weitverbreitet. Neben der Dokumentation des Inhalts der User-Story (**Wer** möchte **was** vom System zu **welchem Zweck**) repräsentiert eine User-Story ein Kommunikationsversprechen. Darüber werden wir bei Bedarf im Detail reden!

Die Vermittlung mit User-Storys funktioniert grob nach folgendem Schema:

- Formulieren der Anforderungen in User-Storys
- Diskussion mit den empfangenden Personen der Anforderungen anhand der formulierten User-Story
- Gemeinsames Zustimmen über die Inhalte der User-Story

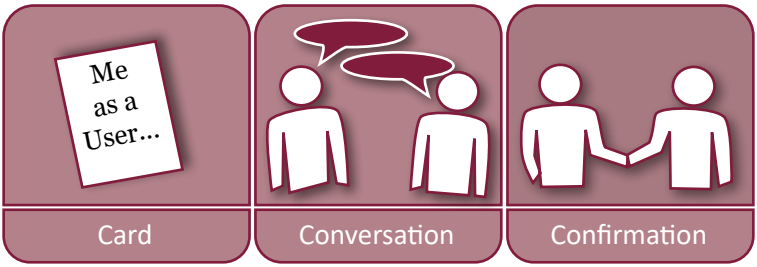



Abbildung 5.1: 3-C-Modell nach Ron Jeffries

Da User-Storys im Allgemeinen nur sehr kleine Funktionalitäten beschreiben, werden Sie im Laufe eines Entwicklungsvorhabens sehr viele dieser User-Storys erzeugen. Um den Überblick zu behalten, können diese dann mit Hilfe eines Story-Mappings zueinander in Beziehung gesetzt werden.

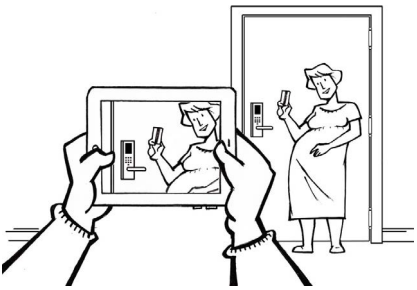
## Prototypen

Prototypen sind in vielen Bereichen der Systementwicklung einsetzbar, sie eignen sich zum Ermitteln von Anforderungen, zu deren Prüfung, aber vor allem auch zum Vermitteln. So vielfältig wie ihre Einsatzmöglichkeiten sind, so unterschiedlich sind auch ihre Ausprägungen. Wir haben gute Erfahrungen mit den folgenden Arten von Prototypen in Systementwicklungen gemacht:

- Wireframe: Eine schematische Darstellung der Elemente einer Benutzungsoberfläche.
- Funktionaler Prototyp: Eine vorläufige Implementierung einer Funktion ohne z.B. auf die Performanz oder Darstellung der Ergebnisse Rücksicht zu nehmen.
- Mock-up der Oberfläche Eine Erweiterung der Wireframes, da hier schon das Design der Oberfläche erkennbar wird.

### Bilder

Ein Bild sagt mehr als tausend Worte... aber sind die mehr als tausend Informationen relevant für die Vermittlung Ihres Wissens? Genau vor diesem Dilemma stehen Sie, wenn Sie Bilder für die Vermittlung von Wissen verwenden. Vielen Menschen fällt es leichter, Ideen und Wünsche in einem Bild festzuhalten als sie nur sprachlich zu explizieren – somit ist ein Bild für sie eine optimale Erzählhilfe. Zudem zeigen Forschungsergebnisse, dass Bilder wesentlich besser gemerkt werden können [Wolfe10], was gerade bei der Wissensvermittlung der entscheidende Faktor ist. Natürlich ist die Verwendung von Bildern eingeschränkt, da sich nicht jeder Aspekt gut in einem Bild ausdrücken lässt. Aber gerade, wenn es um räumliche Anordnungen, Anmutungen, also die Teile eines Systems, die man sehen und bildlich darstellen kann, geht, helfen Bilder enorm bei der Wissensvermittlung.



Wenn wir hier von Bild sprechen, dann meinen wir eine informelle visuelle Darstellung, also z. B. eine Zeichnung des Hauses auf Papier, um über die Positionierung der Überwachungskameras zu diskutieren. Es ist keine formale oder semiformale Darstellung wie ein Architekturmodell oder ein Ablauf, der in einem UML-Diagramm notiert wird.

### Gemeinsame Artefakte erstellen

Eine weitere Möglichkeit Anforderungen zu vermitteln, ist das gemeinsame Erstellen zusätzlicher Artefakte. Besonders gut eignen sich nach unserer Erfahrung das gemeinsame Erstellen von Testfällen zu den Anforderungen. Es ist aber auch denkbar, dass andere Artefakte, zum Beispiel Architektur- oder Designdokumente oder Bedienungsanleitungen, gemeinsam erstellt werden. Gerade wenn Sie im Systems-Engineering tätig sind, könnte die Erstellung von Architekturdokumenten sinnvoll sein (siehe Kapitel 9 - „Systems-Engineering“). Suchen Sie sich Artefakte aus, die sowieso erstellt werden müssen, um keinen weiteren Aufwand zu generieren und die erstellten Artefakte in Folgeprozessschritten nutzen zu können.

## 5.2 Anforderungen mittels Dokumentation vermitteln


Kommen wir nun zu der Vermittlungstechnik, die noch immer den höchsten Verbreitungsgrad hat: Die Dokumentation. Wir finden diese natürlich in einem klassischen Beauftragungsverhältnis in Form eines Lastenhefts oder Pflichtenhefts, aber auch zur Kommunikation zwischen Abteilungen innerhalb einer Organisation. Darüber hinaus wird diese Technik fast unumgänglich, wenn man Anforderungen zur Wiederverwendung zur Verfügung stellen möchte.

## Die Repräsentationen: Natürlichsprachliche vs. modellbasierte Anforderungen

Reden wir über Dokumentation, so kommen wir an den Begriffen natürlichsprachlich und modellbasiert nicht vorbei. Beide Arten der Dokumentation haben Vor- und Nachteile.

Natürlichsprachliche Dokumentation zeichnet sich dadurch aus, dass kein Erlernen einer Notation notwendig ist, da natürliche Sprache jedem verständlich ist. Natürliche Sprache eignet sich darüber hinaus für die Dokumentation aller Arten von Anforderungen. Dabei ist jedoch Vorsicht geboten, denn natürliche Sprache ist oftmals mehrdeutig oder missverständlich.

Eine modellbasierte Anforderungsdokumentation hingegen eignet sich sehr gut, um das System aus verschiedenen Perspektiven isoliert zu betrachten, z. B. der rein strukturellen Sicht auf die zu verarbeitenden Begriffe/Informationen/Daten, die funktionale Sicht auf Workflows/Systemabläufe oder die zustandsorientierte Verhaltensperspektive, die unter anderem Systemreaktionen auf Ereignisse beleuchtet.

Durch die kompakte und für den geübten Leser eindeutig verständliche Darstellung können Missverständnisse vermieden werden. Die Nachteile liegen jedoch auf der Hand – die entsprechende Notation muss erst von allen Beteiligten gelernt und verstanden werden.

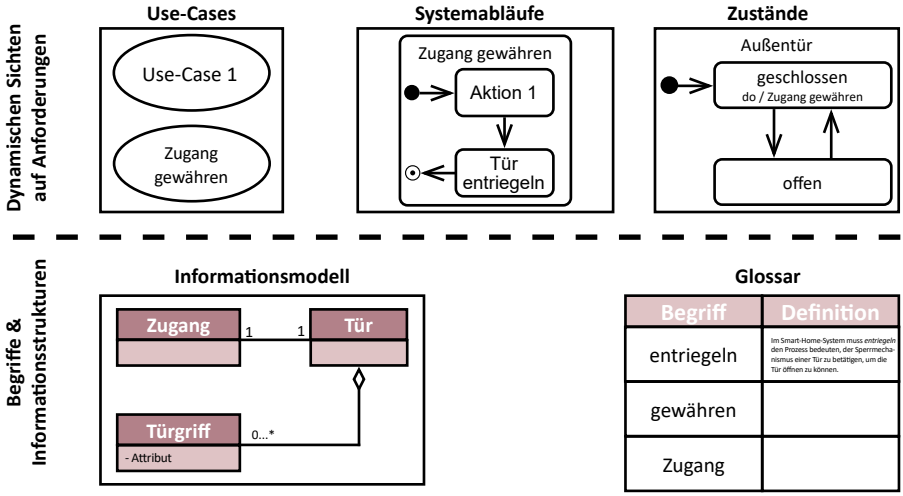



Abbildung 5.2: Sichten auf Anforderungen

GIF www.sophist.de/bre/k5a2/

Um komplexe Sachverhalte in Modellen darzustellen, ist auch meist eine Diagrammart allein nicht ausreichend. Modelle sind auch nicht universal einsetzbar. Ein verbreitetes Vorgehen in der Praxis ist, modellbasierte und natürlichsprachliche Anforderungen zu kombinieren, um die Vorteile beider Formen zu nutzen.

## Ein Musterkandidat für die natürlichsprachliche Dokumentation – der FunktionsMASTER

Aus dem Bereich der natürlichsprachlichen Dokumentation bieten wir Ihnen hier einen unserer Meinung nach einfach einzusetzenden Ansatz an. Die SOPHIST-Satzschablone, auch SOPHIST Requirements-Template, ist ein Bauplan, der die Struktur eines einzelnen Anforderungssatzes festlegt. Die Struktur der einzelnen Anforderungen wird dadurch vereinheitlicht und man kann bereits auf den ersten Blick feststellen, ob wichtige Bestandteile einer Anforderung fehlen. Gerade beim Spezifizieren in einer Fremdsprache kann ein vorgegebenes Anforderungsgerüst dabei helfen, Unsicherheiten zu überwinden.

Die Anwendung der Satzschablone ist leicht erlernbar und reduziert unerwünschte sprachliche Effekte durch die vorgegebene Syntax bereits beim Schreiben einer Anforderung. Verglichen mit willkürlich formulierten Prosaanforderungen steigt somit die Anforderungsqualitätbereits bei der ersten Anwendung deutlich. Die SOPHIST-Satzschablone für funktionale Anforderungen ist inzwischen ein fester Bestandteil der meisten Requirements-Engineering-Prozesse in Unternehmen und wird von uns als FunktionsMASTER bezeichnet.

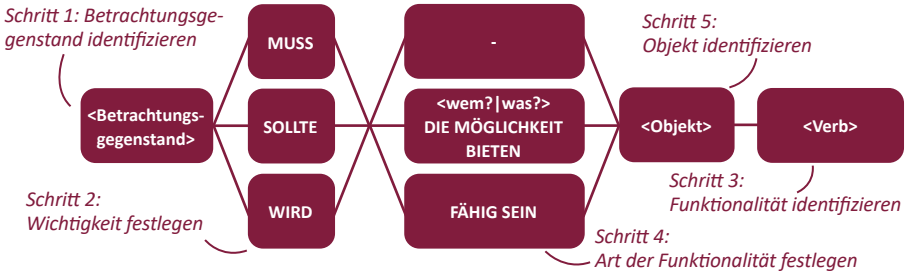



Abbildung 5.3: Der FunktionsMASTER

GIF www.sophist.de/bre/k5a3/

Aufgrund von Praxiserfahrungen haben wir das Konzept weiter entwickelt und weitere Schablonen definiert, um neben den funktionalen Anforderungen auch nichtfunktionale Anforderungen und Bedingungen abzudecken. Details zu unseren neuen Schablonen finden in unserem Buch Requirements-Engineering und -Management [Rupp20].

Unsere Projekterfahrung hat gezeigt, dass die ersten Schritte im Umgang mit den Schablonen erst mal gar nicht so leicht fallen. Das Schreiben der ersten Anforde-

rungen nach Schablone fühlt sich unbeholfen an und es entsteht kurzfristig mehr Aufwand. Der Aufwand kommt aber meist daher, dass man Wissensbestandteile, die dank der Schablone im Satz landen, erst mal reflektieren muss und man damit sofort eine bessere Anforderung erzeugt. Zudem werden Sie anfangs darüber nachdenken, welche Schablone Sie wählen müssen. Schreiben Sie gerade eine Anforderung an das System, das eigenständig etwas tun soll? Ist der Benutzer oder eine Schnittstelle involviert? Wenn Sie es schaffen, sich durch die ersten Stunden durchzukämpfen, so werden Sie feststellen, dass Schablonen ein effektives Mittel sind, schnell und professionell gute Anforderungen zu formulieren. Wir stehen Ihnen hier in einem Training, Workshop oder mit Beratung gerne unterstützend bei.

# 6. Anforderungen verwalten – Requirements-Management

Das Verwalten von Anforderungen, Requirements-Management oder auch Anforderungsmanagement genannt, umfasst die Prozesse, die Sie im Rahmen aller anderen Haupttätigkeiten des RE und der weiteren Verwendung der Anforderungen unterstützen. In den folgenden Abschnitten geben wir Ihnen einen Einblick in die Welt des Requirements-Managements (kurz: RM) und Tipps aus der Praxis, in welchem Umfang Sie welche Tätigkeiten des Requirements-Managements durchführen sollten.

Doch warum sollte man sich überhaupt mit dem Verwalten von Anforderungen beschäftigen?

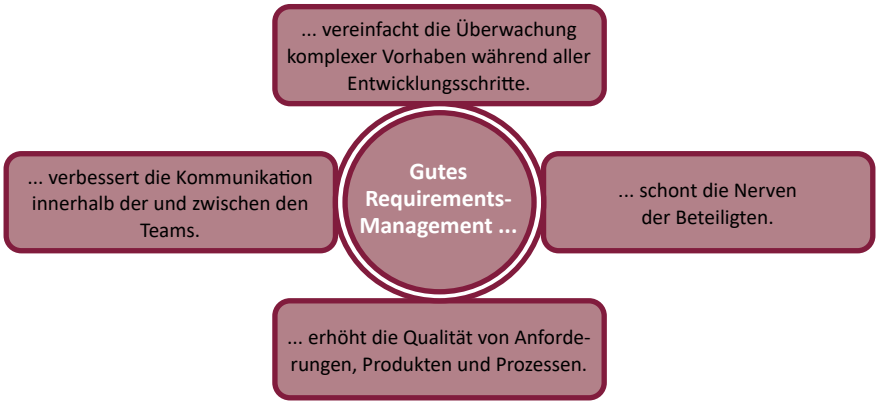



Abbildung 6.1: Gutes Requirements-Management zahlt sich aus

Die abstrakten Gründe aus Abbildung 6.1 haben wir mit zwei prominenten Vertretern konkretisiert.

## Anforderungen ändern sich

Da sich Anforderungen im Laufe der Systementwicklung häufig ändern, ist es notwendig, dass Sie sich in Ihrer Anforderungssammlung zurechtfinden. Änderungen reichen dabei von kleinen Ausbesserungen wie der Korrektur von Rechtschreibfehlern bis hin zu komplexen Änderungen, die umfangreiche Überarbeitungen ganzer Abschnitte Ihrer Spezifikation umfassen. Ein strukturiertes Vorgehen, wie Sie mit solchen Change-Requests umgehen, sollte in Ihrem RE-Konzept festgehalten werden.

## Anforderungen werden weiterverwendet

Behalten Sie stets im Hinterkopf, dass Anforderungen nie zum Selbstzweck gesammelt werden, sondern dass Stakeholder, z. B. Entwickler oder Tester, Ihre Anforderungen lesen, verstehen und damit arbeiten müssen. Als Requirements-Engineer müssen

Sie folglich dafür sorgen, dass diese umfassenden Informationen in der Anforderungssammlung (Anforderungsspezifikation und/oder Product-Backlog) übersichtlich aufbereitet sind.

## 6.1 Wie viel Requirements-Management ist sinnvoll?

Die Bedeutung des Requirements-Managements innerhalb des Entwicklungsprozesses steht in einem direkten Zusammenhang mit den Rahmenbedingungen Ihres Entwicklungsvorhabens. Zwar gibt es keine genaue Vorgabe über den Aufwand, den Sie für RM einplanen sollten, doch die folgenden Einflussfaktoren können Ihnen dabei helfen, den Aufwand für RM einzuschätzen:

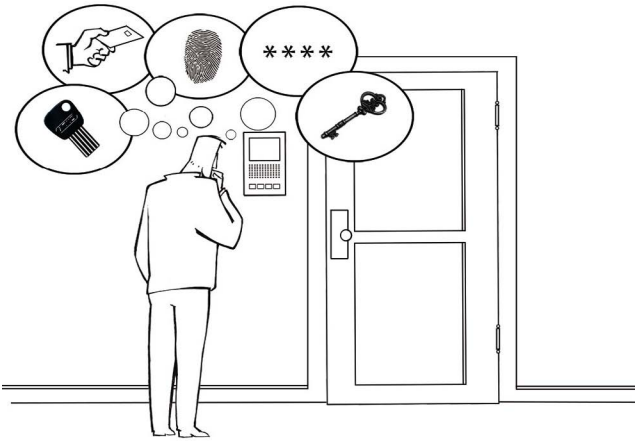
- Anzahl/Umfang der Anforderungen und der weiteren Informationen
- Zu erwartende Lebensdauer des Produkts
- Höhe der Änderungsrate
- Anzahl der Beteiligten am Prozess
- Verfügbarkeit der Stakeholder
- Qualitätsanspruch an das System
- Grad der Wiederverwendung
- Komplexität/Kompliziertheit des Entwicklungsprozesses
- Heterogenität der Stakeholdermeinungen
- Anzahl der zu entwickelnden Releases
- Vorhandene Toollandschaft für das Requirements-Management
- Vorgehensweise im Projekt
- Externe Vorgaben (Normen, Zertifizierungsprozessen oder Unternehmensrichtlinien)

Aus unserer Erfahrung heraus empfehlen wir Ihnen, bei der Analyse Ihrer Rahmenbedingungen nicht in eine Lethargie des „Das war schon immer so“ zu verfallen. Versuchen Sie stattdessen zwei Dinge herauszufinden:

- Welche Rahmenbedingungen sind unveränderlich und auf welche können Sie Einfluss nehmen?
- Wie oder wodurch können Sie Ihre Rahmenbedingungen modifizieren?

Denn in der Praxis zeigt sich, dass bereits geringfügige Änderungen an bestehenden Rahmenbedingungen eine große Erleichterung bzw. Verbesserung bewirken können. Im Folgenden gehen wir auf einige der wichtigsten Aspekte des Requirements-Managements näher ein.

## 6.2 Versionierung und Baselines




Da sich Anforderungen im Laufe der Systementwicklung und teils auch darüber hinaus ändern, hat sich das Einführen einer Versionierung von Anforderungen als geeignetes Mittel herausgestellt. So kann auch zu einem späteren Zeitpunkt noch nachvollzogen werden, wie sich die Anforderungen im Laufe der Zeit verändert haben.

Beim Erzeugen einer neuen Version einer Anforderung wird die Anforderung zunächst kopiert. Die alte Anforderung bleibt bestehen und wird mit der neuen Version verknüpft. Diese neue Version erhält darüber hinaus eine neue Versionsnummer. Die alte Anforderung wird in der Historie Ihrer Anforderungssammlung eingetragen. Die neue Version kann nun von Ihnen bearbeitet werden. Durch dieses Vorgehen wird sichergestellt, dass keinerlei Informationen verloren gehen. Viele speziell für RM entwickelte Tools unterstützen Versionierung. Dies ist einer der Gründe für ein professionelles RM-Tool.

Versionierung hilft Ihnen darüber hinaus auch dabei, Releases und Anforderungsänderungen zu planen. Dafür legt man einen unveränderbaren Stand an Anforderungen fest, auf den man auch zu einem späteren Zeitpunkt wieder zurückgreifen kann. Diese Auswahl an Anforderungen bezeichnet man als Konfiguration. Wenn eine Konfiguration alle Anforderungen für ein Release umfasst, spricht man statt von einer Konfiguration von einer Basislinie oder Baseline. Geben Sie jeder Konfiguration und Baseline eine eindeutige Bezeichnung, um sie identifizieren zu können.

Um herauszufinden, welche Anforderungen zu einer Konfiguration oder einer Baseline gehören, benötigen Sie ein Konzept für die Traceability.

## 6.3 Traceability

**Definition Traceability nach SOPHIST [RUPP20]:**

Traceability ist die Fähigkeit, Verbindungen und Abhängigkeiten zwischen Informationen, welche während der Analyse, Entwicklung, Wartung, Weiterentwicklung bis hin zur Entsorgung oder Ablöse eines Systems anfallen, jederzeit nachvollziehen zu können.

Mit einer Traceability nach der obigen Definition können Sie beispielsweise bei Änderung einer Anforderung herausfinden, welche weiteren Anforderungen von dieser Änderungbetroffen sind, welche Anforderungen für die Entwicklung einer Systemfunktionalität benötigt werden oder welche Testfälle diese Anforderungen abdecken.

Traceability schafft die Grundlage für effektives und qualitativ hochwertiges Requirements-Management, da sie folgende Aspekte unterstützt:

- Nachweisbarkeit: Wurden alle Ziele, vereinbarten Anforderungen, Testfälle etc. umgesetzt? Wurden alle Vorgaben eingehalten?
- Identifikation von Abhängigkeiten: Welche Auswirkung hat eine Änderung einer Anforderung auf weitere Entwicklungsartefakte?
- Wiederverwendung: Welche Artefakte aus dem Entwicklungsprozess werden in anderen Projekten verwendet?
- Nachvollziehbarkeit und Übersicht: Wie hat sich das System entwickelt und verändert? Mit welchen Aufwänden muss bei einer Fehlerbehebung gerechnet werden?

Definieren Sie in einem Verfolgbarkeitsmodell, welche Traceability Sie benötigen, wer diese zu welchem Zeitpunkt und in welcher Weise pflegt und wie Sie diese Traces im Verlauf Ihres Vorhaben oder darüber hinaus nutzen wollen. Denn unsere Erfahrung zeigt, dass Sie nicht unterschätzen sollten, wie viel Aufwand die Erstellung und Pflege der Traceability zwischen den von Ihnen verwalteten Informationen bedeuten kann.

## 6.4 Change- und Release-Management

Häufige und komplexe Änderungen an Systemen erfordern ein durchdachtes Vorgehen, um alle anstehenden Prozesse abzudecken – sei es das Sammeln von Änderungswünschen, die Planung von Releases, oder auch der Roll-Out der durchgeführten Änderungen. Die passenden Methoden lassen sich der Disziplin des Change- und Release-Managements zuordnen.

Einen Überblick über den Zusammenhang gibt die folgende Abbildung.

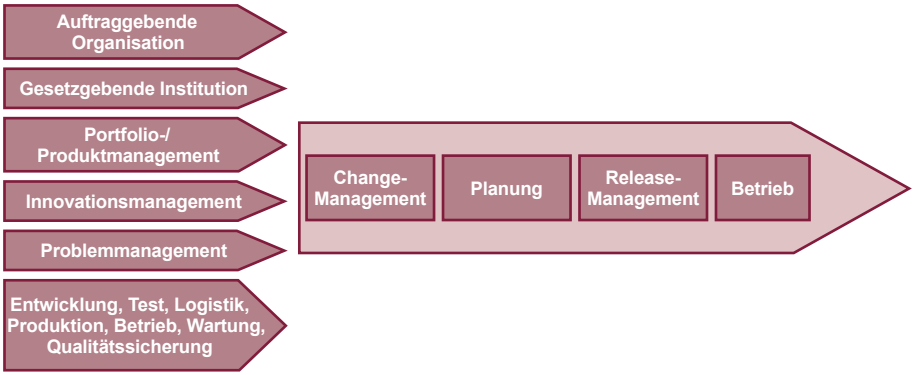



Abbildung 6.2: Change- und Releasemanagement

Im linken Drittel sehen Sie die potenziellen Quellen für Änderungen – die Erklärungen der einzelnen genannten Quellen finden Sie in Kapitel 2 - „Was ist Requirements-Engineering?“. In den rechten beiden Dritteln sehen Sie dann den Prozess einer Änderung vom Change-Management über die Umsetzung bis hin zur Übernahme der Änderung in den Betrieb des Systems.

## Change-Management

Das Change-Management steuert den Lebenszyklus aller Änderungen mit dem Ziel, die Änderungen kontrolliert in den Entwicklungsprozess einzusteuern. Zu den Aufgaben des Change-Managements gehören

- Auswirkungsanalysen durchführen,
- Änderungen beurteilen,
- Änderungen priorisieren,
- Änderungen planen,
- und die Annahme bzw. Ablehnung von Änderungen kommunizieren.

## Release-Management

Sobald die Planung, Terminierung und Steuerung von Builds und Tests sowie das Integrieren in bestehende Systeme ansteht, kommt das Release-Management ins Spiel. Eine besondere Rolle hat dabei der Release-Manager inne, der die Einhaltung aller Termine kontrolliert.

Ist die Umsetzung abgeschlossen, muss das neue System zum Kunden. Der Inhalt des Releases kann sehr variieren – so kann eine neue Produktversion entstehen, wenn viele Innovationen umgesetzt wurden oder es kann auch eine Rückrufaktion

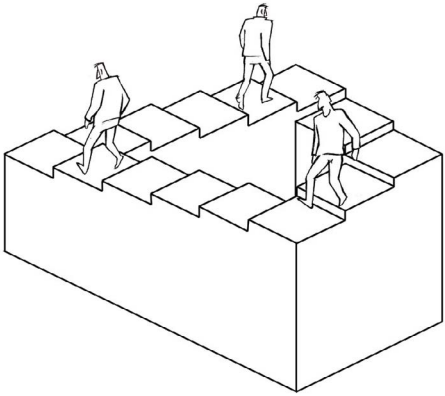
starten, wenn Sie einen kritischen Fehler beseitigt haben. Vergessen Sie dabei nicht Ihre Servicehotline, Ihr Schulungspersonal, Ihre Handbuchautoren und alle weiteren betroffenen Rollen rechtzeitig mit den notwendigen Informationen, z. B. in Form von Release-Notes, zu versorgen, um diese auf die Änderungen vorzubereiten.

# 7. Einführung eines verbesserten Requirements-Engineerings

In jedem Entwicklungsvorhaben findet Requirements-Engineering statt, allerdings nicht immer effektiv, effizient und zur Zufriedenheit aller Beteiligten. Falls Sie das Requirements-Engineering bei sich verbessern wollen, dann sind Sie hier genau richtig. Dabei können Änderungen den Prozess, die Methoden und auch das Tooling oder alles gleichzeitig betreffen. Egal, ob Sie von einem wasserfallartigen zu einem agilen Requirements-Engineering wechseln wollen oder dem Requirements-Engineering den letzten Schliff verpassen wollen, sollten Sie Verbesserungen immer systematisch durchführen.

## 7.1 Veränderungen in einer Organisation

Der britische Sozialphilosoph Herbert Spencer prägte den Ausdruck vom Überleben des am besten Angepassten (Survival of the Fittest). Dies gilt nicht nur für jeden einzelnen Menschen, sondern auch für Unternehmen. Märkte verändern sich ebenso wie die Ansprüche von Kunden und Partnern. Wer mit dem technischen Fortschritt nicht mithält, wird bald der Konkurrenz hinterherhinken. Gerade ein Thema wie die Digitalisierung hat das Potenzial, Märkte stark zu verändern. Deswegen müssen sich die Systeme an sich und auch, wegen neuer Erkenntnisse und Möglichkeiten, die Entwicklungsprozesse und damit auch das Requirements-Engineering anpassen.



Jedoch bedeutet eine Anpassung auch die Infragestellung des Althergebrachten, denn jede Veränderung bedeutet Aufgabe von etwas Gewohntem und von der Sicherheit, die Bekanntes bietet. Folglich ist Veränderung immer von Angst, Bedenken oder Zweifeln begleitet.

Tief im Menschen verwurzelt ist das Bedürfnis nach Sicherheit. Etwas Neues, das das Vertraute ersetzen soll, wird oft als Bedrohung wahrgenommen. Unsere Erfahrung zeigt, dass gerade die Angst vor Versagen und die Scheu, etwas Neues zu erlernen und dabei Fehler zu machen, allgegenwärtig ist. Für eine erfolgreiche Einführung brauchen Sie zu Beginn ein Problembewusstsein, den Handlungsbedarf und eine Änderungsbereitschaft. Ist dies alles vorhanden, dann können Sie sich auf die Suche nach einer Verbesserungsidee machen und, wenn diese erprobt wurde, einführen.

## 7.2 Eine Einführung ist ein Projekt!

Solche Verbesserungen müssen aber sorgsam in eine Organisation eingeführt werden. Gehen Sie dies genauso geordnet wie ein Entwicklungsprojekt an. Dabei ist es unerheblich, ob Sie die Einführung eines neuen Vorgehens, von Methoden oder eines neuen Tools planen oder diese erst erfinden und erproben müssen.

Wir erleben es immer wieder, dass die Beteiligten bei der Analyse der Leitplanken, die an manchen Stellen bestehen und den Lösungsraum einschränken, überrascht sind. Nicht nur, dass unbekannte Leitplanken identifiziert werden. Genauso häufig werden Leitplanken berücksichtigt, die gar nicht existieren.

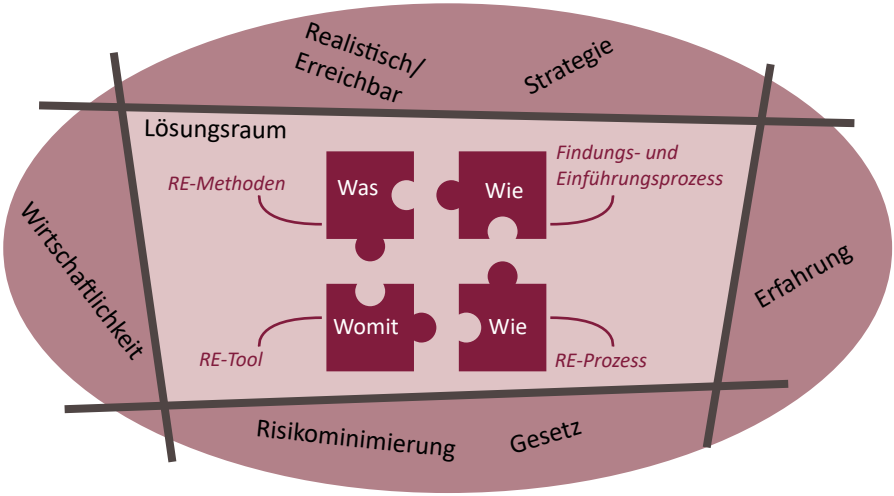



Abbildung 7.1: Leitplanken einer RE-Einführung

## 7.3 Veränderungen agil gestalten

Für die Verbesserung von Requirements-Engineering gibt es unterschiedliche Vorgehensweisen. Unser Ansatz, den wir häufig in unserer Arbeit verwenden und der für viele Ausgangssituationen geeignet ist, basiert auf einer agilen Abwicklung.

Wir haben gelernt, dass sich die Wünsche unserer Kunden und Kundinnen ständig ändern und dass wir dieser Dynamik mit Agilität erfolgreich begegnen können.

Manche dieser Veränderungen sind nicht inhaltlich, sondern betreffen Leitplanken oder neue Technologien und führen zu einer geänderten Arbeitsweise.

Aber auch dieser Dynamik können wir begegnen, indem wir unser agil entwickeltes Vorgehen anpassen.

Damit dieser Ansatz funktioniert, gehen wir von ein paar Grundannahmen aus, die innerhalb der gesetzten Leitplanken liegen müssen.

- Das Team selbst muss die Hoheit über seine Arbeitsweise haben, also innerhalb der gesetzten Grenzen eigenständig entscheiden dürfen.
- Allen Beteiligten muss klar sein, dass das Ausprobieren der neuen oder geänderten Methoden das Ziel hat, etwas auszuprobieren und zu lernen – und Ergebnisse eventuell auch wieder verworfen werden.

Um ein solches Vorgehen durchzuführen, gibt es zwei Aspekte, die unserer Erfahrung nach die Erfolgswahrscheinlichkeit enorm steigert. Erstens sollten bei der Teamauswahl nur Teams berücksichtigt werden, die nicht nur Änderungsbereitschaft signalisieren, sondern vor allem auch Handlungsbedarf haben. Zweitens sollte mindestens ein „Elvis" im Team sein, denn keiner hat mehr NachahmerInnen, jede/r will wie er sein. Der Erfolg von Imitationslernen, Akzeptanz und Vertrauensvorsprung erleichtert die spätere Verbreitung und Einführung des neuen Vorgehens. Im Folgenden werden wir die einzelnen Schritte des dargestellten Prozesses erläutern (Siehe **Abbildung 7.2**).

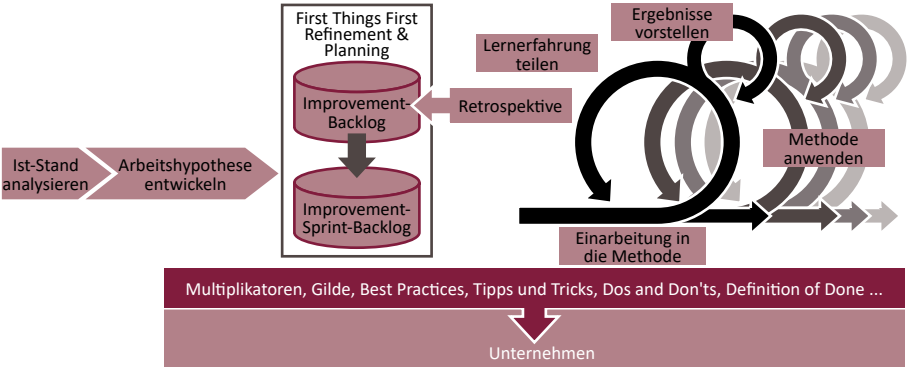



Abbildung 7.2: Überblick über ein agiles Vorgehen für Veränderungen

Am Anfang wird der **Ist-Stand analysiert** und eine **Arbeitshypothese entwickelt**. Hier liegt der Fokus auf der aktuellen Arbeitsweise des Teams. Untersucht werden Requirements-Engineering-Methoden und -Tools und vor allem, welche Reibungsverluste im Team und nach außen, z. B. zu den Stakeholdern, existieren. Die daraus entwickelten Verbesserungsideen werden in einem Improvement-Backlog abgelegt. Nach der Übernahme in den Improvement-Sprint-Backlog erfolgt die **Einarbeitung in die Methode**. Ob das ganze Team oder nur die am Experiment Teilnehmenden sich einarbeiten, entscheidet das Team. Während die **Methode angewendet** wird, sammelt das Team Erfahrungen im Anwenden der Methode unter realen Arbeitsbedingungen. Das Team kann hier auch experimentieren, ob es die Neuerung am Stück (Workshop-Charakter) oder kontinuierlich ausprobieren möchte. Während dieser Zeit bekommen die Teilnehmenden jede methodische Hilfe, die sie anfragen. Dies kann eine weitere Schulung, ein Review der Ergebnisse, ein Coaching, Sparringspartner ... sein. Ob es gerade Bedarf gibt und wie die am Experiment Teilnehmenden vorankommen, fragen Sie am besten täglich ab – im Daily. Um die gemachten

Erfahrungen zu bewerten, werden den Stakeholdern die **Ergebnisse vorgestellt** und deren Feedback abgefragt. Wie leicht oder aufwändig die Anwendung der Methode war, erfährt das Team von den Experimentierenden, wenn diese ihre **Lernerfahrung teilen**. Damit kann das Team in der **Retrospektive** über den Prozess und die Ergebnisse reflektieren. Die aus der Retrospektive abgeleiteten Maßnahmen füllen dann wieder den Improvement-Backlog.

Nicht nur wenn das Team für sich eine Verbesserung gefunden hat, auch über die Erfahrungen mit dem Vorgehen kann das Team anderen berichten und damit den Rollout teilweise selbst in die Hand nehmen.

Wenn nicht nur ein Team bzw. nicht nur ein Thema verbessert werden soll, dann können Sie dieses agile Vorgehen mit den üblichen Frameworks wie z. B. SAFe [SAFe] skalieren.

## 7.4 Arbeitspakete einer Einführung

Um den Erfolg Ihrer Einführung abzusichern, lohnt es sich die wichtigsten Schritte mittels der folgenden Konzepte zu planen und diese auszuarbeiten.

- Marketingkonzept: Planen Sie wen Sie von der Idee begeistern wollen.
- Konzept zur Wissensvermittlung: Strukturieren Sie den systematischen Wissensaufbau.
- Pilotierungskonzept: Finden Sie die Kriterien, wann was durch welches Pilotprojekt erprobt wird.
- Migrationskonzept: Durchdenken Sie, wie Sie mit Bestehendem (z. B. vorhandenen Spezifikationen) umgehen.

Wichtig bei all den Konzepten ist, dass Sie die Durchführung der Maßnahmen auch überwachen und mittels Metriken den Erfolg messen. Beispielsweise lohnt es sich bei einem Marketingkonzept vorab ein Ziel zu definieren, wie viele Menschen Sie mit den Informationen erreichen wollen. Prüfen Sie in regelmäßigen Abständen, wie viele der avisierten Menschen Sie bereits von den neuen Ideen überzeugt haben. Nur so wissen Sie, ob die durchgeführten Maßnahmen auch erfolgreich waren.

# 8. Requirements-Engineering und Agilität

Das Wort Agilität ist heutzutage in aller Munde und derzeit aus dem Bereich der Systementwicklung nicht wegzudenken. Daher müssen wir uns natürlich die Frage stellen: Wie sieht es denn mit dem Requirements-Engineering in agilen Systementwicklungen aus?

Zunächst mal können wir aus unserer Erfahrung heraus klar sagen, dass auch in agilen Vorgehensweisen Requirements-Engineering betrieben wird. Vieles wird nur anders genannt und vor allem zu anderen Zeitpunkten erledigt. Allerdings kommen im Agilen häufig auch zusätzliche Aufgaben auf einen Requirements-Engineer zu, die in der klassischen Systementwicklung meistens von anderen Rollen bearbeitet werden.

Es gibt verschiedene agile Vorgehensweisen bzw. Frameworks. Das bekannteste Framework ist dabei Scrum. Typisch für alle Vertreter agiler Vorgehen ist, dass das Produkt Schritt für Schritt entwickelt wird. In vielen Iterationen werden kleine Inkremente des Produkt entwickelt, welche am Ende ein großes Ganzes ergeben. Ein wichtiger Grundgedanke ist dabei, dass nach jeder Iteration ein Ergebnis erzielt werden soll, welches den Stakeholdern gezeigt werden kann und diese dazu Feedback abgeben können. Dieses Feedback muss natürlich in die Anforderungen für die zukünftigen Iterationen einfließen.

Damit wir uns nicht zu Beginn zu viele Gedanken über detaillierte Anforderungen machen, welche erst sehr spät umgesetzt werden sollen und sich daher durch das viele Feedback noch ändern werden, fahren wir in der Agilität den Ansatz des Just-in-time Requirements-Engineerings. Das bedeutet, dass wir uns um die Ausarbeitung der Anforderungen kümmern, die kurz vor der Realisierung stehen.

Dadurch entstehen neue Herausforderungen. Zum Beispiel müssen wir uns stärker mit Abhängigkeiten zwischen Anforderungen beschäftigen, um nicht in einer Iteration etwas zu fordern, was im Konflikt zu Anforderungen vorheriger Iterationen steht. Um mit dieser Herausforderung umgehen zu können, achten wir auf geschicktes Zerlegen der Anforderungen, um möglichst keine Abhängigkeiten zu haben (Stichwort Independent aus dem INVEST-Prinzip, siehe **Abschnitt 2.2 - „Qualität von Anforderungen“**) und nutzen Techniken aus der klassischen Welt (z. B. Modelle), um die Zusammenhänge sichtbar zu machen.

## Die Requirements-Engineer-Rollen in der Agilität

Im agilen Umfeld existiert meistens die Bezeichnung Requirements-Engineer nicht. Allerdings müssen die Aufgaben trotzdem erledigt werden. Denn auch eine agile Entwicklung kann nicht darauf verzichten, die Wünsche der Stakeholder zu ermitteln, um dieses Wissen aufbereitet an die Entwicklung zu vermitteln. In der Agilität obliegen diese Aufgaben meistens dem Product-Owner. Allerdings kann dieser auch Unterstützung durch andere Fachkräfte bekommen, welche dann Proxy-Product-Owner, Business Analysten oder manchmal tatsächlich auch Requirements-Engineers

genannt werden. Wie viele von Ihnen wahrscheinlich bereits wissen, hat ein Product-Owner mehr Aufgaben als die typischen Requirements-Engineering-Tätigkeiten. Ein Product-Owner kümmert sich auch um Themen wie Release-Planung, Priorisierung und Entscheidungen über den Umfang des zu erstellenden Produkts. Das sind typischerweise keine klassischen Requiremtents-Engineering Aufgaben. Wie Sie sehen, kommt auf den Requirements-Engineer im Agilen noch einige zusätzliche Arbeit zu.

## Die Requirements-Engineering-Tätigkeiten in der Agilität

In der vorliegenden Broschüre haben wir bereits die Haupttätigkeiten des Requirements-Engineering erörtert. Wir sprechen dabei von Wissen ermitteln, gute Anforderungen herleiten, Anforderungen vermitteln und Anforderungen verwalten. Zum Verwalten wollen wir später kommen und zunächst mal nur die ersten drei Haupttätigkeiten betrachten.

**Wissen ermitteln:** Diese Tätigkeit lässt sich sehr leicht in der agilen Welt wiederfinden. Denn dort machen wir Requirements-Engineers bzw. Product-Owner eigentlich nichts anderes als in der klassischen Welt. Als Product-Owner machen wir uns auch Gedanken über Ziele, Stakeholder oder Sytem- Kontextabgrenzung.

Allerdings kommen durch die Agilität neue, tolle Techniken (wie z. B. Vision Box, News from the future) hinzu, die wir aber auch gerne in klassisch durchgeführten Entwicklungsvorhaben einsetzen. Einer Herausforderung stellen wir uns aber als Product-Owner sehr häufig. Zwar können wir als Product-Owner sehr wohl eine System- und Kontextabgrenzung machen, aber die wird über einen langen Zeitraum noch sehr „instabil“ sein, da wir uns nur Schritt für Schritt mit den Anforderungen beschäftigen.

**Gute Anforderungen herleiten:** Auch in der Agilität müssen wir aus den Aussagen der Stakeholder gute Anforderungen herleiten. Allerdings sehen die Anforderungen aufgrund der anderen Arbeitsweise häufig anders aus. Wir arbeiten üblicherweise nicht darauf hin, alle Anforderungen komplett ausdetailliert niederzuschreiben.

Sondern wir beschreiben die Anforderungen eher ansatzweise, um dann mit anderen Personen darüber zu sprechen. Es werden eher gröbere User-Storys erstellt. Da wir für die Aussage, was gute Anforderungen sind, entsprechende Qualitätskriterien verwenden, nutzen wir im agilen entsprechend andere als im klassischem. Häufig nehmen wir das INVEST-Prinzip als Qualitätsziel unserer Anforderungen (siehe **Abschnitt 2.2 - „Qualität von Anforderungen“**).

Gerade in dieser Tätigkeit ist das Schneiden der User-Storys ein wichtiges Thema. Da der Erfolg sehr stark vom richtigen Schneiden abhängt, haben sich aus unseren

Erfahrungen ein paar Schneidetechniken und Kriterien herauskristallisiert, nach denen geschnitten werden kann, um gute Ergebnisse zu erzielen.

**Anforderungen vermitteln:** Gerade in dieser Haupttätigkeit gibt es viele Unterschiede zwischen agiler und nicht agiler Entwicklung. In der agilen Welt werden Anforderungen vor allem durch Gespräche in Refinement-Meetings vermittelt. Wir haben also nicht mehr die klassischen Dokumente, in denen die Anforderungen nachgelesen werden können, sondern wir haben eine Sammlung von Anforderungen in einem Product-Backlog. Diese Anforderungen können unterschiedlich beschrieben sein und mit unterschiedlichen Techniken vermittelt werden. Man sollte sich dabei nicht nur auf eine unstrukturierte Diskussion beschränken. Wir nutzen häufig Techniken wie zum Beispiel Storytelling, gemeinsames Erstellen von Testfällen oder Videos zu Vermittlung der Anforderungen.

## Die Anforderungssammlung in der Agilität

In klassischen Vorgehen wird üblicherweise eine Anforderungsspezifikation erstellt (z. B. ein Lastenheft). Dieses enthält zum Ende der Anforderungsanalyse alle Anforderungen in der gewünschten Qualität und dem benötigtem Verfeinerungsgrad. In der Agilität verwenden wir als Anforderungssammlung ein Product-Backlog. Dort finden sich alle zu einem Zeitpunkt bekannten Anforderungen. Aber eben nicht alle Anforderungen für ein Produkt. Auch erfüllen nicht alle Anforderungen die vergebenen Qualitätskriterien und den benötigten Verfeinerungsgrad. Das gilt nur für die Anforderungen (oder besser Product-Backlog Items), die in einer nahegelegenen Iteration umgesetzt werden sollen. Neben diesen Eigenschaften gibt es noch einen weiteren großen Unterschied zur Anforderungsspezifikation. Während die Anforderungen in einer Anforderungsspezifikation nach fachlichen Zusammenhängen gegliedert und sortiert sind, haben wir in einem Product-Backlog eine Sortierung nach der Priorität. Also die Anforderungen, die ganz oben im Product-Backlog stehen, haben aktuell die höchste Priorität. Dabei kann es durchaus vorkommen, dass diese keine fachlichen Zusammenhänge haben.

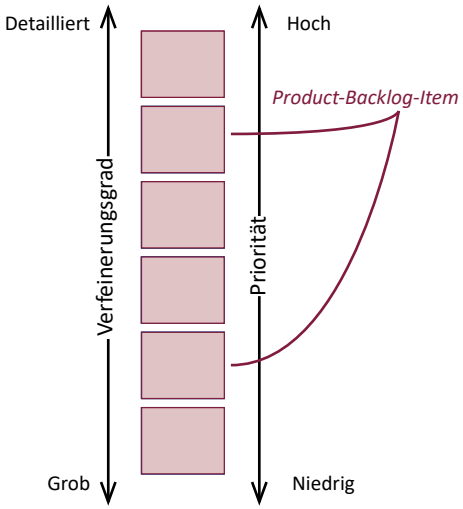
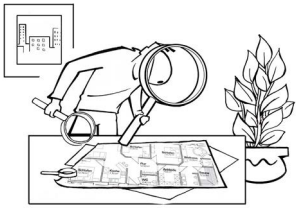



Abbildung 8.1: Das Product-Backlog

Das erschwert natürlich das inhaltliche Arbeiten an den Product-Backlog Items. Das lässt sich aber durch verschiedene Techniken bewältigen. Wir nutzen sehr erfolgreich verschiedene Story Maps oder arbeiten neben dem Product-Backlog auch noch mit weiteren Dokumentationsformen. Wir haben also viele Möglichkeiten, das Requirements-Engineering in der agilen Welt gewinnbringend zu gestalten.

# 9. Systems-Engineering

Im Rahmen eines Systems-Engineerings-Vorgehens gewinnt ein geordnetes Requirements-Engineering entscheidend an Bedeutung. Nicht nur weil durch die Systemanalyse auf Systemebene der Grundstein für die Gesamtentwicklung gelegt wird, sondern auch, da sich das Arbeiten mit Anforderungen auf allen Ebenen des betrachteten Systems wiederfindet.



Bevor wir diese Aussagen genauer vorstellen, müssen wir einige zentrale Begriffe im Systems-Engineering definieren.

## 9.1 Definitionen und Ziel

Die erste Definition beschäftigt sich mit dem Begriff des Systems-Engineerings an sich.

**Definition Systems-Engineering nach INCOSE [INCOSE20]:**

Systems-Engineering ist ein interdisziplinärer Ansatz und Mittel, um erfolgreiche Systeme zu realisieren. Es konzentriert sich auf die Definition von Kundenbedürfnissen und erforderlicher Funktionalität in einem frühen Stadium des Entwicklungszyklus, die Dokumentation von Anforderungen, die anschließende Design-Synthese und Systemverifizierung unter Berücksichtigung des vollständigen Problems.

Schauen wir uns ein paar Begriffe in dieser Definition genauer an.

Mit dem „vollständigen Problem“ ist die Betrachtung des gesamten Lebenszyklus des Systems gemeint. Wir müssen also nicht nur Anforderungen betrachten, die den Einsatz des Systems adressieren. Vielmehr müssen schon zu Beginn der Entwicklung auch die Bedürfnisse der Produktion, des Transports, Lagerung, und Installation bis hin zur Entsorgung des Systems einfließen.

Ein weiterer zentraler Begriff in der Definition ist natürlich das System.

**Definition System nach SOPHIST [RUPP20]:**

Ein System besteht aus mehreren Teilen und das sichtbare Verhalten und die Eigenschaften ergeben sich aus dem Zusammenspiel dieser Teile.

Daraus leitet sich auch eine weitere wichtige Tätigkeit eines Systems-Engineerings ab: Die Zerlegung des Systems in seine Komponenten. Dies werden wir im Weiteren als „Systemarchitektur“ bezeichnen, in der wiederum Anforderungen eine Rolle spielen werden.

Viele der Systeme, mit denen wir in unseren Projekten konfrontiert werden, weisen noch zwei weitere Eigenschaften auf:

- Sie sind so komplex, dass die Zerlegung über mehrere Ebenen betrachtet wird. Ein Teil des Systems wird als Subsystem betrachtet und wiederum durch einen Architekturschritt in seine Teile zerlegt.
- Die Komponenten des Systems stammen aus unterschiedlichen Gewerken wie z. B. Software, Elektronik, Mechanik etc.

Daraus folgt, dass sich die Entwicklung des Systems (oder eines Teils) in ein größeres System einbettet. Somit folgen die Anforderungen an ein System auf beliebiger Zerlegungsebene aus der Systemarchitektur der darüber liegenden Systemebene.

Betrachten wir zum Schluss noch einen weiteren Begriff aus der obigen Definition. Systems-Engineering soll ein „erfolgreiches" System liefern. Für uns ist ein System erfolgreich, wenn die folgenden Ziele erreicht wurden:

- Das Projekt ist „in Time & Budget", was bedeutet, dass die Abschätzungen für den Entwicklungsaufwand und die Entwicklungszeit eingehalten wurden.
- Die abgeschätzten Herstellungskosten wurden eingehalten. Aber auch die Qualität bei der Produktion des Systems kann sichergestellt werden.
- Das System weißt die benötigten Eigenschaften auf. Diese können sich von den anfänglichen Stakeholder-Anforderungen unterscheiden, da sich unter anderem das darüber liegende System ändern kann.

Der erste Punkt wird in der nachfolgenden Abbildung aufgegriffen. Dort wird die Entwicklungszeit und der Entwicklungsaufwand mit und ohne Einsatz von Systems-Engineering-Methoden dargestellt.

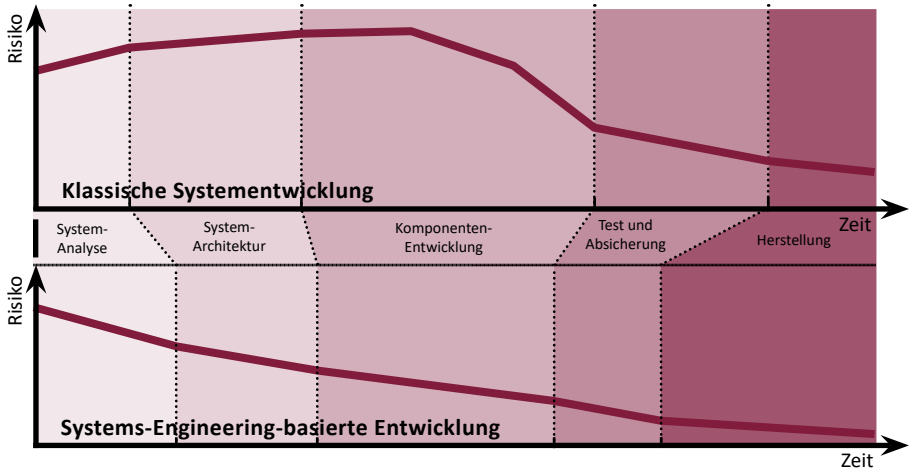



Abbildung 9.1: Kosten und Risiken vor und nach der Einführung von Systems-Engineering

Durch den Einsatz von mehr Aufwand in der Systemanalyse erwartet man insgesamt eine Zeit- und damit auch eine Geldeinsparung bei einer gleichbleibende Qualität. Als zusätzlicher Effekt wird das Risiko früh minimiert und nicht erst während der Realisierungsphase.

## 9.2 Einbettung des Requirements-Engineerings

Aus der Abbildung 9.1 folgt, dass in einer geordneten Systementwicklung eine qualitativ hochwertige Systemanalyse für das betrachtete System durchgeführt werden sollte. Wie im vorigen Abschnitt angedeutet, kann ein System aus mehreren Subsystemen bestehen, für die wir dann ein ähnliches Vorgehen vorschlagen.

Dieser Gedanke lässt sich am Besten in einer Abbildung darstellen.

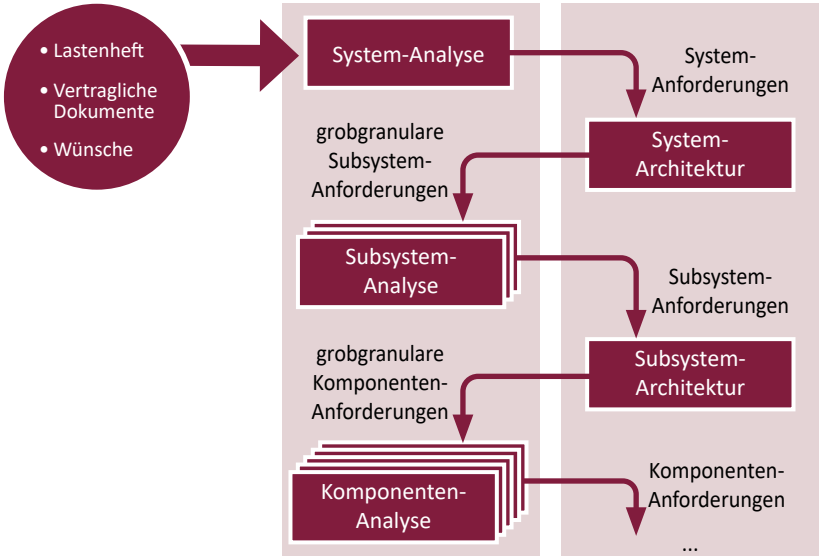



Abbildung 9.2: Abfolge von Analyse und Architektur auf mehreren Ebenen

Die Systemanalyse erzeugt aus den Eingaben die zu entwickelnden Systemanforderungen. Diese sind Eingaben in einen Systemarchitekturschritt, in dem unter anderem zwei, für diesen Kontext wichtige Schritte durchführt werden:

- Die Subsysteme des Systems werden identifiziert.
- Aus den Systemanforderungen werden Anforderungen an die Subsysteme abgeleitet.

Die Anforderungen an ein Subsystem drücken dabei aus, wie dieses Subsystem an der Realisierung der Systemanforderungen beteiligt ist. Damit können diese grobgranularen Anforderungen als Lastenheftanforderungen für das Subsystem angesehen werden.

Bei der Realisierung des Subsystems ist es nun wiederum die erste Aufgabe, die an es gestellten Anforderungen zu analysieren, um belastbare Subsystemanforderungen zu erhalten. Diese können dann in einer Subsystemarchitektur auf die jeweiligen Komponenten verteilt werden.

Ohne Berücksichtigung von einigen Sonderfällen setzt sich diese Abfolge von Analyse- und Architekturschritten so weit nach unten fort bis

- eine Komponente gefunden wurde, die außerhalb der verantwortlichen Systemorganisationseinheit entwickelt wird (z. B. durch einen Zulieferer) und/oder
- eine Komponente gefunden wurde, die ganzheitlich einer Domäne (Mechanik, Software, …) zugeordnet werden kann.

Mit diesem Vorgehen können wir eine durchgängige Betrachtung der Anforderungen auf den unterschiedlichen Systemebenen gewährleisten.

Viele der bisher vorgestellten Ansätze, Methoden und Techniken werden Sie auf jeder Systemebene verwenden können. Bei der Entwicklung von komplexen, technischen Systemen können jedoch noch andere Techniken zum Einsatz kommen, die z. B. aus Gesetzen und Normen folgen.

So werden Sie vielleicht eine FMEA (Failure Mode and Effects Analysis [Werdich12]) einsetzen müssen, um Schwachstellen in der Realisierung zu finden. Aus diesen gefundenen Risiken werden Sie dann Anforderungen ableiten, die Sie auf der betrachteten Systemebene in die Entwicklung einbringen.

Weiterhin kann Ihnen eine FuSi (Funktionale Sicherheit, [ISO 26262]) neue Eingaben in den Analyseprozess auf einer beliebigen Systemanalyse mit einer HaRA (Hazard and Risk Analysis) und einem funktionalen und einem technischen Sicherheitskonzept liefern.

Gerade in sicherheitskritischen Bereichen wird noch viel mehr von einem Analyseprozess gefordert. Dies betrifft zum Beispiel die Verfolgbarkeit der Anforderungen zu den Eingaben, der Realisierung und den Tests oder auch die Überprüfungen, die für die Anforderungen durchgeführt werden müssen.

Für die Definition eines Vorgehens im Requirements-Engineering bei solchen Systemen ergeben sich insgesamt die folgenden Treiber, die möglichst alle erfüllt werden sollten:

- Prozessuale Standards und Normen, die sich zum Teil ergänzen und widersprechen
- Qualität der Anforderungen, wie sie für die weitere Entwicklung oder Vergabe an einen Zulieferer benötigt wird
- Zeit und Aufwand, gegeben durch das Projekt

Diese Treiber in Zusammenhang mit der im Kapitel 7 motivierten „vorsichtigen“ Veränderung machen es nicht einfach, die geeigneten Maßnahmen für ein kommendes Systems-Engineering-Projekt zu definieren. Unsere Erfahrung zeigt, dass auch hier eine Sicht von außen von Experten im methodischen Bereich sehr hilfreich ist und Ihnen die Sicherheit geben kann, die richtigen nächsten Schritte zu gehen.

# 10. Requirements-Engineering bei Smart Ecosystems und als Treiber der digitalen Transformation

Prof. Dr. Key Pousttchi vom Lehrstuhl für Wirtschaftsinformatik und Digitalisierung der Universität Potsdam definiert die digitale Transformation wie folgt:

**Definition digitale Transformation nach Pousttchi [Pousttchi17]:**

Der Begriff digitale Transformation bezeichnet erhebliche Veränderungen des Alltagslebens, der Wirtschaft und der Gesellschaft durch die Verwendung digitaler Technologien und Techniken sowie deren Auswirkungen.

Die Auswirkungen sind dabei weitreichend. Sie ziehen sich durch nahezu alle Bereiche unseres alltäglichen Lebens. Geschäftsmodelle und Wertschöpfungsketten verändern sich drastisch. Kommunikationsmedien und damit auch die Art zu kommunizieren verändern sich. Letztendlich verändern sich damit die Arbeitswelt und auch unsere Lebensentwürfe. Diese neuartigen Systeme, aber auch die neu betroffenen Stakeholder, haben eine signifikante Auswirkung auf unsere Arbeitswelt als Requirements-Engineer.

Das hier in der Broschüre verwendete Beispiel des Smart-Home-Systems, in dem viele Einzelsysteme (Videoüberwachung, Heizung, Jalousiesteuerung, Zugangskontrolle,…) miteinander kommunizieren und interagieren, ist ein solches Smart Ecosystems. Das Smart-Home-System selbst stellt dabei die Kommunikationsplattform für die in ihm enthaltenen Systeme zur Verfügung. Sie können solche Ökosysteme beliebig skalieren und beispielsweise das Smart-Home-System als Bestandteil eines übergeordneten Ökosystems, einer sogenannten „Smart Rural Area“, betrachten. Derartige Systeme werden meist nicht von einem Unternehmen entwickelt, sondern entstehen aufgrund der Interaktion unterschiedlicher Systeme im gleichen Kontext oder Umfeld. Das Unterscheidungsmerkmal zwischen „normalen“ und smarten Ökosystemen liegt darin, dass bei smarten Systemen zu jeder Zeit weitere Systeme in das Ökosystem aufgenommen oder auch herausgenommen werden.

## Auswirkungen von Smart Ecosystems auf das Requirements-Engineering

Eine spezielle Herausforderung, die sich in unseren Projekten wiederholt offenbart hat, stellt die System- und Kontextabgrenzung für ein System innerhalb eines Smart Ecosystems dar. Die Schwierigkeit besteht zum einen darin, dass Sie eine hohe potenzielle Anzahl an Interaktionspartnern und damit Schnittstellen erwartet und zum anderen, dass Sie zum Zeitpunkt der Entwicklung und Implementierung noch nicht eindeutig und vollständig bestimmen können, mit welchen Systemen oder Personen Ihr System interagieren wird.

Diese Problematik ist nicht gänzlich neu. Auch in serviceorientierten Systemen existiert diese Herausforderung. Das Stichwort hier lautet: Schnittstellenverträge bzw. Quality-of-Service-Agreements oder Service-Level-Agreements, durch die sichergestellt wird, dass das aufrufende System eine definierte (Daten-)Qualität aufweist und sich dadurch zur Nutzung des Services berechtigt.

Eine weitere Auswirkung besteht darin, dass die Systeme innerhalb eines Smart Ecosystems sich ergänzen und als Verbund Funktionen zur Verfügung stellen können, die die Einzelsysteme isoliert voneinander nicht realisieren können. Eine mögliche Herangehensweise dafür ist, dass Sie sich nicht nur auf Ihren Betrachtungsgegenstand beschränken, sondern Ihren Blick auch über den Tellerrand richten. Dies gelingt Ihnen, indem Sie die Abstraktionsebenen, die über Ihrem konkreten Betrachtungsgegenstand liegen, analysieren. Stellen Sie sich vor, Sie entwickeln ein Steuergerät innerhalb eines Fahrzeugs. Es kann für Sie nun hilfreich sein neben dem Steuergerät auch das Gesamtfahrzeug zu betrachten, um Abhängigkeiten und das Zusammenspiel der Einzelteile im Großen und Ganzen transparent zu machen (siehe auch **Kapitel 9 - „Systems-Engineering“**). Eine Empfehlung aus unserer Erfahrung heraus ist, dass Sie dabei auch die Geschäftsprozess-Ebene in Ihre Analyse mit einbeziehen. Durch diese Betrachtung gelingt es Ihnen auch andere mögliche Einsatzszenarien oder Anwendungsmöglichkeiten für Ihr zu entwickelndes System zu identifizieren.

## Mögliche Lösungsansätze zur Analyse im Rahmen von Smart Ecosystems

Ein möglicher Ansatz zum verbesserten Requirements-Engineering bei einem Smart Ecosystem ist die Einbettung des Arbeitens mit den Anforderungen in eine agile Entwicklung. Dies liegt vor allem daran, dass Smart Ecosystems eine hohe Komplexität aufweisen, zum Zeitpunkt der Entwicklung viele Unschärfen bestehen und binnen kürzester Zeit eine Vielzahl an Änderungen an Gegebenheiten und Rahmenbedingungen auftreten können. Hier sollten Sie Agilität aber auch in einem größeren Rahmen einführen, leben und perfektionieren können. Eine spannende Herausforderung für Sie als Requirements-Engineer, die viel Erfahrung erfordert.

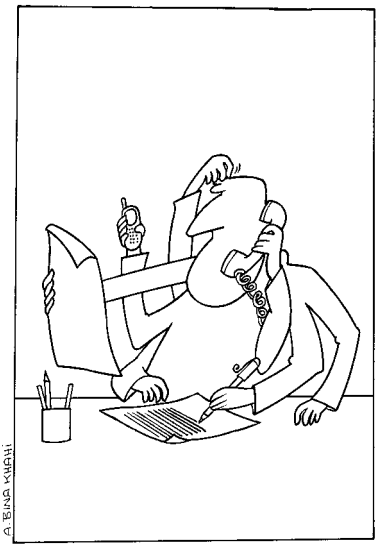
Ein zweiter möglicher Ansatz ist model-based Systems-Engineering. Model-based Systems-Engineering eignet sich im Kontext von Smart Ecosystems, da es durch die Dekomposition des Gesamtsystems und der Betrachtung unterschiedlicher Sichten auf das zu entwickelnde System die Menge der Informationen und Daten portioniert, damit die Komplexität/Kompliziertheit beherrschbar wird, sodass das Gesamtsystem verständlich und entwickelbar wird.

Fundierte Kenntnisse im Bereich Requirements-Engineering und Modellierung sind hier das A und O, aber auch Erfahrung im Umgang mit komplexen Systemen.

Auch der Einsatz künstlicher Intelligenz wird das Requirements-Engineering beeinflussen. Die Spezifikation eines KI-Systems unterscheidet sich nur geringfügig von der eines herkömmlichen Systems. Der signifikante Unterschied ist der Betrachtungsgegenstand. Wo man bei einem klassischen System die Funktionsweise, das System-

verhalten und die technische Realisierung in den Mittelpunkt stellt, steht bei einem KI-System die Spezifikation des Trainingsprozesses sowie der Qualität der hierfür verwendeten Daten im Vordergrund. Durch diese Herangehensweise verlagern Sie die Komplexität aus der Spezifikation in das KI-System. Sie geben lediglich das Ziel und die Kriterien für den produktiven Einsatz vor, ohne die Funktionsweise im Detail durchdringen zu müssen.

## Digitale Transformation im Requirements-Engineering



Für die **Ermittlung** von Wissen stehen Sie vor der Herausforderung, dass durch neue Technologien Wissen an unterschiedlichen Stellen 0vorhanden ist. Das bedeutet, dass vor allem die Selektion der Anforderungsquellen damit zunehmend an Priorität gewinnt. Zwei Werkzeuge, welches wir Ihnen wärmstens empfehlen können, insofern Sie Stakeholder haben, die für Sie nur schwer oder gar nicht zugreifbar sind, sind Personas und Empathy Maps. Darüber hinaus ermöglichen neue Technologien Ihnen als Requirements-Engineer eine spannende Ergänzung Ihrer Arbeitsweise. Wir empfehlen Ihnen bei der Ermittlung von Anforderungen z. B. zusätzlich auf den Einsatz von CrowdRE-Methoden, Living Labs, Design Thinking oder anderen Co-Creation-Ansätzen zurückzugreifen. Gönnen Sie sich hier zumindest kurzfristig ExpertInnen, die Erfahrung mit derartigen Techniken haben.

Ein weiterer Aspekt ist die Veränderung der Arbeitswelt, in der mobiles Arbeiten Normalität wird. Die räumliche Trennung erschwert den Einsatz von Ermittlungstechniken, die die physikalische Präsenz der Teilnehmer erfordern. Hier helfen spezielle Tools, die sie virtuell zusammenrücken lassen, oder eben der Einsatz anderer Ermittlungstechniken, die nicht unter diesem Problem leiden.

Bedingt durch die Digitalisierung werden zudem viele manuell durchgeführte Prozesse automatisiert. Sie sollten also genug Zeit und Aufwand in Ihre Geschäftsprozessanalyse investieren, um auch undokumentierte manuelle Prozesse adäquat digitalisieren zu können.

Auf die Hautptätigkeit **gute Anforderungen erzeugen** wirkt sich die digitale Transformation insofern aus, als dass Assistenzsysteme Sie beim Erzeugen guter Anforderungen unterstützen können. So können beispielsweise Texterkennungssysteme, um Sie auf Qualitätsmängel in Ihren Anforderungen hinzuweisen oder Bilderkennungssoftware, die Diagrammskizzen in syntaktisch korrekte Diagramme überführt, eingesetzt werden.

Auf die **Vermittlung** von Anforderungen wirkt sich vor allem die zunehmende digi-

tale Kommunikation aus. Sie wird besonders die Zusammenarbeit zwischen dem Requirements-Engineer und seinen Stakeholdern verändern. Früher hingen Product-/Sprint-Backlogs, Roadmaps oder Kanban-Boards mit Haftnotizen an Wänden. Heute werden solche Artefakte digitalisiert, da nicht alle Beteiligten vor Ort sind. Meetings, die davon leben mit haptischen Objekten zu arbeiten, werden schwerer durchführbar. Hier ist der Einsatz von Virtual oder Augmented Reality hilfreich.

Falls Ihre Stakeholder verteilt sitzen und eine direkte Zusammenarbeit beim Verbessern der Anforderungen nicht möglich ist, so sind auch hier kreative, toolgestützte Arbeitsweisen die Rettung. Schwergewichtige, umfangreiche Dokumente werden vermutlich so nicht mehr bestehen, da der Änderungsaufwand durch regelmäßige und kontinuierliche Auslieferung und Bereitstellung (engl. Continuous Delivery) und einen raschen technologischen Fortschritt kaum mehr handhabbar wäre – außer Sie haben Ihre Modelle durch ein professionelles Systems-Engineering und ein gute Tool im Griff. Inzwischen greifen wir zur Dokumentation und Vermittlung häufig auf Videos zurück. Dadurch entsteht eine neue spannende REpräsentation im Requirements-Engineering, deren Potenzial sie heben sollten.

Beim **Verwalten** von Anforderungen gewinnt vor allem das Change-Management an Bedeutung, da sich die Gesellschaft als Ganzes mit mehr Technologien auseinandersetzt und diese aktiv nutzt. Binden Sie dementsprechend Ihre Stakeholder aktiver mit ein und rechnen Sie mit vielen Change-Requests.

# 11. Business-Analyse oder Requirements-Engineering oder beides?

In vielen Unternehmen gibt es die Rolle oder Jobbezeichnung Requirements-Engineer nicht. Dort gibt es AnforderungsmanagerIn, Business-Consultant, Business-AnalystIn usw. Worin genau der Unterschied zwischen diesen Jobbeschreibungen liegt, ist, abhängig vom Unternehmen, mehr oder weniger weniger klar beschrieben. In der Literatur wird zumindest zwischen zwei Begriffen klar unterschieden: Requirements-Enginering und Business-Analyse.

Was sich hinter dem Begriff Requirements-Engineering verbirgt, haben wir Ihnen in Kapitel 2 bereits erklärt. Unter Business-Analyse versteht das International Institute of Business Analysis (kurz: IIBA) folgendes:

**Definition Business-Analyse nach Interational Institute of Business Analysis [IIBA]:**

Business-Analyse ist eine Tätigkeit, Veränderungen in einem Unternehmen zu ermöglichen, indem der Unternehmensbedarf definiert und Lösungen empfohlen werden, die den Stakeholdern Wert bringen. [BABOK®v3]

Bereits in dieser Erklärung können Sie erkennen, dass es einen Zusammenhang zwischen Business-Analyse und Requirements-Engineering gibt: es werden „Unternehmensbedarfe definiert […], die den Stakeholdern Wert bringen“. Das hört sich ziemlich nach Anforderungen an, oder? D.h. kurz gesagt: Requirements-Engineering ist eingebettet in die Business-Analyse. Doch was bedeutet dies im Detail?

Die Business-Analyse ist eine sehr breit gefasste Disziplin, die viele einzelne Tätigkeiten umfassen kann. Ganz prinzipiell müssen Business-AnalystInnen die Strategien, Ziele, Geschäftsprozesse, usw. eines Unternehmens verstehen und sie mit den Anforderungen des Marktes kombinieren, um Produkte/Verbesserungen zu entwickeln, die den betriebswirtschaftlichen Zielen des Unternehmens entsprechen. Dazu kann man die vier im folgenden beschriebenen Bereiche festlegen.

## Arbeitsschritte planen und steuern

Eine der häufigsten Tätigkeiten, die wir in der Business-Analyse durchführen, ist die Planung und Steuerung aller Business-Analyse-Aktivitäten. Dabei zeigt unsere Erfahrung, dass häufig ein einfacher PDCA-Zyklus implementiert wird. Zum Planen (P) zählen für uns bekannte Dinge wie beispielsweise Stakeholderanalyse, Aufsetzen der Anforderungsverwaltung und Planung der einzelnen Arbeitsschritte. Die Arbeitsschritte werden dann durchgeführt (D) und das Ergebnis dokumentiert. Jetzt kann überprüft (C) werden, ob die Schritte das angestrebte, erwünschte Ergebnis

gebracht hat und ob es effektiv war. Abschließend werden ggfs. Anpassungen (A) an den Arbeitsschritten abgeleitet, um evtl. Defizite im Vorgehen auszugleichen. Die Planung und Steuerung der Arbeitsschritte ist ein Bereich, der Sie während der gesamten Entwicklung begleitet.

## Business-Case erstellen

Um eine Verbesserung zu erzielen oder ein neues Produkt zu entwickeln, müssen Sie zunächst verstehen, was benötigt wird. Dazu ermitteln wir mit Ihnen Potenziale, leiten Geschäftsanforderungen ab, erheben Lösungsalternativen und bewerten diese. Dies können Sie ganz klassisch mittels Machbarkeitsstudien, Wirtschaftlichkeitsprüfungen, usw. machen. Unsere Projekterfahrung zeigt jedoch, dass heutzutage auch hier verstärkt auf agile Techniken zurückgegriffen wird: Product Canvas mit Minimum Viable Product (MVP), Vision Statements, Product Box, usw., die im Zuge eines Elevator Pitchs vorgestellt werden.

## Requirements-Engineering

Haben Sie es geschafft? Haben Sie ihr Management von Ihrer Produktidee/Ihrem Business-Case überzeugt? Dann startet das Requirements-Engineering. Dabei ist es egal, ob Sie in einem klassischen oder agilen Vorgehensmodell arbeiten. Sie ermitteln Detailinformationen zur ausgewählten Lösungsalternative, leiten daraus Anforderungen ab, dokumentieren, priorisieren und vermitteln diese an Ihr Entwicklungsteam. Wie Sie das am besten machen, erfahren Sie in unseren Broschüren, Büchern, auf unserer Webseite oder von uns persönlich.

## Entwickelte Lösung einführen

Zum Abschluss folgt der wichtigste Teil: die Begleitung der Einführung der Lösung in ihre Zielumgebung. Besonders wenn Sie eine neues Software für die Optimierung Ihrer Geschäftsprozesse entwickelt haben oder gar wenn Ihre Lösung Personal ersetzt, dann müssen Sie unserer Erfahrung nach besonders umsichtig vorgehen. Es reicht nicht aus, nur die technischen Voraussetzungen für die Integration Ihrer Software in die Produktivumgebung zu erfüllen – das wäre einfach. Nein, die größte Gefahr lauert darin, dass Ihre Stakeholder die Lösung aufgrund von Existenzangst oder der Angst vor Veränderung ablehnen. Der Schlüssel liegt hier in einem gut geplanten, organisatorischen Veränderungsprozess (siehe auch **Kapitel 7 - „Einführung eines verbesserten Requirements-Engineerings“**) der alle Betroffenen/Beteiligten mitnimmt.

Sie sehen, Business-Analyse und Requirements-Engineering sind eng miteinander verzahnt. Lassen Sie sich dazu von uns beraten!

# 12. Videos im RE

Im Rahmen der RE-Tätigkeiten werden im Allgemeinen viele Anforderungen entstehen, die unterschiedlich dokumentiert bzw. vermittelt werden sollen. Diese Anforderungssammlung kann dabei so komplex werden, dass der Zusammenhang zwischen den Anforderungen, trotz des Einsatzes modellbasierter Dokumentationstechniken oder Story-Maps, verloren gehen kann.

Hier kommen Videos ins Spiel, die es unter Anderem ermöglichen, die vielen kleinen Anforderungen und ihren Zusammenhang verständlicher darzustellen. Aber auch bei der Ermittlung können Videos dadurch helfen, dass das ursprüngliche Problem oder auch der Kontext des neu zu gestaltenden Systems dargestellt wird.

Das Erstellen und Versenden solcher Videos ist dank der modernen Technologie kein Problem mehr. Es gibt günstige bzw. kostenfreie, leicht bedienbare Apps für Smart-Devices, um Videos zu erstellen und zu bearbeiten.

Natürlich sind das Planen, das Erstellen und das nachträgliche Bearbeiten der Videos nicht ganz umsonst. Die Zeit und damit die Kosten, die Sie dafür benötigen, rechnen sich allerdings gegenüber den Ersparnissen bei der Vermittlung der Anforderungen, besonders bei einer großen Anzahl von Stakeholdern und Konsumenten der Anforderungen. Um die Kosten für die Videos so gering wie möglich zu halten, haben wir die wichtigsten Überlegungen und Prinzipien für deren Erstellung und Verwendung kurz zusammengefasst.

## Der PILZ des Videos

Bevor Sie mit dem Videodreh starten, sollten Sie sich genau Gedanken machen, was der Inhalt Ihres Videos sein soll, bzw. welchem Zweck das Video dienen soll. Um diese Überlegungen strukturiert durchzuführen, haben wir den PILZ neu definiert. In PILZ sind vier verschiedene Dimensionen adressiert, die den Inhalt und die Form des Videos (das Drehbuch) maßgeblich bestimmen.

- Phase: In welcher Phase des Requirements-Engineering-Prozesses soll das Video eingesetzt werden. Soll es zum Beispiel die Ermittlung oder die Vermittlung der Anforderungen unterstützen?
- Inhalt: Soll in dem Video ein eher statischer oder dynamischer Inhalt vermittelt werden? Beispiele hierfür sind die Darstellung des Eingangsbereiches des Hauses, in den sich ein Teil des neuen Smart Home Systems einbetten soll (sta-

tisch) oder der aktuell typische Ablauf beim Betreten des Hauses ohne das neue System (dynamisch).

- Lösungsbezug: Wie viel von einer möglichen (fachlichen oder technischen) Lösung wollen Sie darstellen. Sie können die Eingabe eines PIN-Codes an einem Tastenfeld zeigen (technische Lösung) oder Sie verstecken diese Aktion hinter einer eingeblendeten Karte mit der Beschriftung „Authentifizierung“ (lösungsneutral).
- Zeitbezug: Stellt das Video einen Soll- oder einen Ist-Stand dar? In anderen Worten: Wird mit dem Video der, eventuell abstrahierte, Soll-Zustand dargestellt oder spiegelt der Inhalt den momentan gegebenen Ist-Zustand wider?

Für die unterschiedlichen Ausprägungen in den vier Dimensionen haben wir Handlungsempfehlungen definiert, um das Erstellen des Drehbuchs zu unterstützen und damit die Qualität des Videos sicher zu stellen. So sollten Sie zum Beispiel zur Darstellung eines statischen Inhalts zwischen Übersichts- und Detailaufnahmen wechseln, wobei Sie die Details für eine gewisse Zeitdauer ohne Wechsel der Kameraposition zeigen. In [Rupp20] haben wir die vollständige Liste von Ausprägungen der Dimensionen inklusive der jeweiligen Handlungsempfehlungen angegeben.

## Die Einbettung in eine Anforderungssammlung

Ebenfalls Bestandteil unseres Ansatzes ist es zu beschreiben, wie sich Videos mit einer Anforderungssammlung verbinden lassen. Videos im Requirements-Engineering haben in der Regel eine Nutzerperspektive, weswegen sie sich homogen in Dokumentenstrukturen, die als oberste Ordnungsebene Use-Cases verwenden, einbinden lassen.

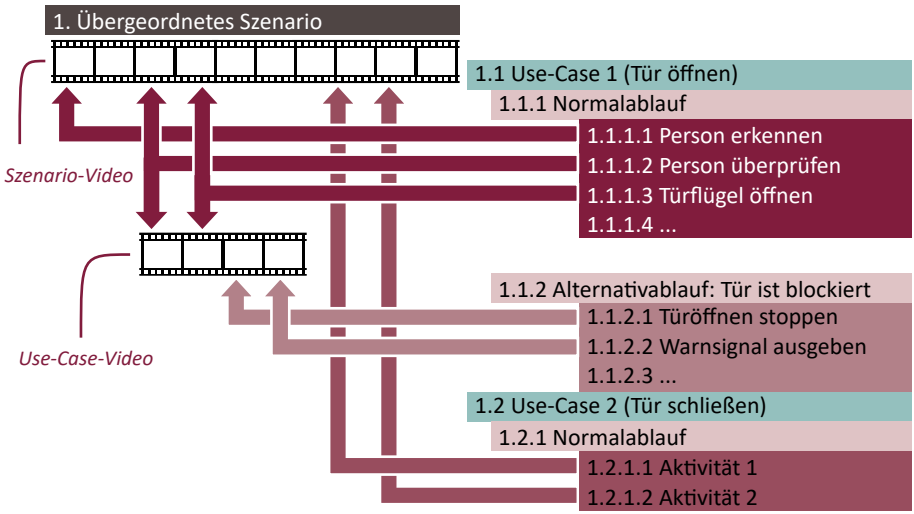



Abbildung 12.1: Detaillierungsgrade von Videos

Bei diesem Ansatz können Sie entscheiden, welche Inhalte Sie auf welcher Abstraktionsebene in verschiedenen Videos darstellen wollen. In Abbildung 12.1 sind unterschiedliche Videos auf Ebene der Use-Cases dargestellt, die unterschiedliche Teile der möglichen Abläufe visualisieren. Diese Use-Case-Videos lassen sich nun auch zur Visualisierung eines übergeordneten Ablaufs zusammen schneiden, so dass Sie ohne viel Aufwand Ihre Anforderungssammlung auf unterschiedlichen Abstraktionsebenen ergänzen können.

## Videos drehen in einem Workshop

Der gestalterische Akt des Erstellens von Videos erfordert eine sehr intensive Auseinandersetzung mit den darzustellenden Inhalten. Unserer Erfahrung nach lässt sich die Erstellung eines Videos sehr gut in einem Workshop durchführen. Bei der Planung und beim eigentlichen Dreh des Videos können Defizite in einem beschriebenen Ablauf auffallen, Details müssen konkretisiert werden und bestehende Anforderungen werden nochmal auf den Prüfstand gestellt, ob sie z.B. nicht doch schon (nicht beabsichtigte) Lösungsanteile enthalten. Bei der Erstellung des Drehbuchs werden Sie also auf eine ganz besondere Art und Weise mit und an den Anforderungen arbeiten, wobei die Sicht von mehreren Beteiligten immer hilfreich ist. Der eigentliche Dreh ist dann immer ein Gruppenereignis, das allen Haupt- und Nebendarstellern lange in Erinnerung bleiben wird.

Neben PILZ haben wir für Sie noch weitere Hilfsmittel für die Durchführung solcher Video-Workshops parat, z. B. Storyboards und Moodboards. Näheres hierzu finden Sie auf unserer Website unter www.sophist.de/re7/kapitel27.

# 13. RE für Produktlinien und -familien

Wir KundInnen haben es gut! Immer und überall können wir aus vielen Alternativen wählen – oder noch besser– unser Produkt individuell kreieren. Damit uns das gelingt, werden wir z. B. durch Konfiguratoren unterstützt. Denn in vielen Bereichen gilt: Je individueller das Produkt, desto besser.

Die wachsende Auswahl und Individualität führen allerdings in der Produktentwicklung zu Komplikation. Um an dieser Stelle den Aufwand seitens der Entwicklung möglichst zu begrenzen, ist besonders die Analyse von Wiederverwendungsmöglichkeiten und Produktähnlichkeiten von hoher Bedeutung. Das Ziel ist, möglichst vielfältige Kundenwünsche mit dem geringstmöglichen Entwicklungsaufwand zu erfüllen. Die meisten Produkte werden heutzutage nicht neu, sondern evolutionär weiterentwickelt. Fast niemand bringt ein einzelnes Produkt, sondern meist gleich eine Produktfamilie oder Produktlinie, in denen sich die Produkte mehr oder weniger voneinander unterscheiden, auf den Markt. Ganz im Sinne von „shift left“: Je früher im Entwicklungsprozess dieser Gegebenheit Rechnung getragen wird, desto höher ist das Einsparpotenzial im gesamten Entwicklungsprozess – idealerweise schon im Portfoliomanagement bei der Definition der Produktstrategie.

## Das Feature-Modell

Wir gehen also von mehreren Produkten aus, die sich in ihren Eigenschaften und Leistungen gleichen, aber eben auch unterscheiden. Diese Eigenschaften oder Leistungen heißen Merkmale oder Features. Mit ihnen lassen sich später Varianten auswählen oder unterscheiden. Für die Dokumentation hat sich das Feature-Modell als Notation bewährt.

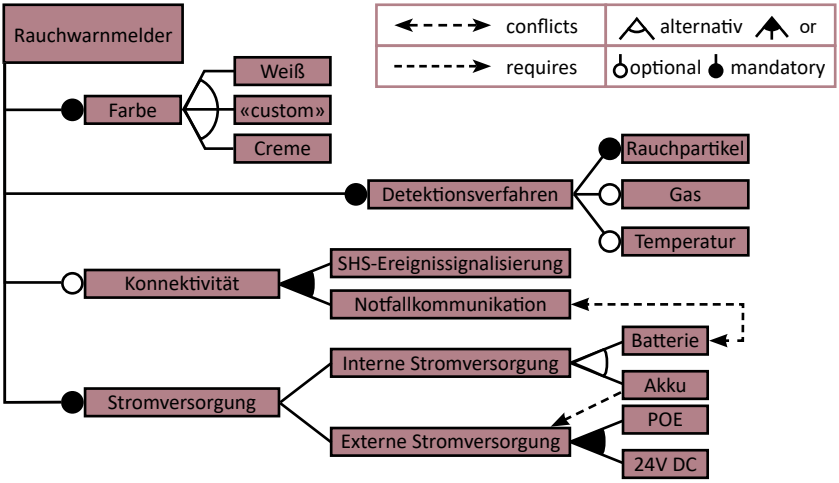



Abbildung 11.1: Feature-Modell des Rauchwarnmelders

Wird dann das Produkt „wie immer“ beschrieben, kann dann jeder Anforderung das passende Feature aus dem Feature-Modell zugeordnet werden. Ihre Anforderungen können Sie mit den Features verlinken oder einfach als ein Attribut Ihrer Anforderungen sehen, mit dem Sie diese Zuordnung herstellen.

| Rauchwarnmelder<br>Use-Cases; Anforderungen | Quelle/Ableitung aus Feature |
|---|---|
| Detektiere Gefahr | Immer |
| - Detektiere Rauchpartikel | Rauchpartikel |
| - Detektiere Gas | Gas |
| - Detektiere Temperatur | Temperatur |
| Signalisiere Event | Immer |
| - Signalisiere Gefahr | Immer |
| - Signalisiere Ereignis | SHS-Ereignissignalisierung |
| Nutze Sprechverbindung | Notfallkommunikation |
| Lade Akku | Akku |
| Stelle Verbindung zur Zentrale her | SHS-Ereignissignalisierung |
| Schalte System ein | Immer |
| Schalte System aus | Immer |

Abbildung 11.2: Zuordnung Anforderung Features

Auf Basis des Feature-Modells mit seinen Regeln lassen sich einfach verschiede Varianten und natürlich auch das Minimum Viable Product (MVP) oder eine Plattform (was allen Produkten gemein ist) bestimmen.

Ziel dabei ist natürlich nicht nur die Anforderungen wiederzuverwenden, sondern auch weitere Entwicklungsartefakte wie z.B. die Architektur oder die Testfälle bis hin zu den Testergebnissen, sofern möglich.

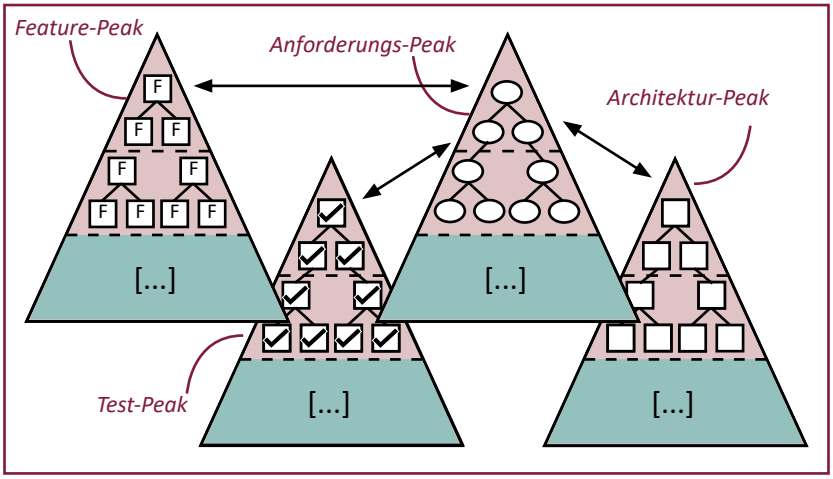



Abbildung 11.3: Peak-Modell

# 14. Fazit

Anforderungen und User-Stories an ein System bilden den Kern einer erfolgreichen Systementwicklung. Mit ihnen steht und fällt ein Projekt, da sie die Voraussetzung für Verträge, eine korrekte Implementierung und Entwicklung, sowie Tests und Abnahmekriterien darstellen. Dies gilt für eine agile Entwicklung genauso wie für eine klassische Entwicklung.

Requirements-Engineering ist ein anspruchsvolles Aufgabengebiet, das viele Möglichkeiten eröffnet. So steigert professionelles Requirements-Engineering die Erfolgsaussichten von Projekten nicht nur im IT-Umfeld, da durch eine sinnvolle Analyse sichergestellt werden kann, dass

- das zu entwickelnde System dem Vorgehen angemessen beschrieben ist.
- die Erfüllung von Zielen durch die zugehörigen Anforderungen oder Stories gesichert wird.
- nur abgestimmte und geprüfte Anforderungen, die der Zielerreichung dienen, umgesetzt werden.

Auch wenn der Aufwand eines dem Vorgehen angepassten Requirements-Engineering hoch erscheint – Sie sparen viel Geld, wenn Sie von Anfang an in die richtige Richtung und gemäß der Kundenwünsche entwickeln. Requirements-Engineering lohnt sich!

Diese Broschüre sollte Ihnen einen Eindruck vermitteln, was zu den grundlegenden Arbeiten eines Requirements-Engineers gehört. Wir konnten dabei natürlich nur einen Teil der umfangreichen Tätigkeit, und das nur grob, vorstellen. Mehr Informationen erhalten Sie in unseren Publikationen, auf unserer Webseite und natürlich in Beratungsgesprächen und Trainings mit uns.

Wenn Sie also von den Vorreitern, Profis und echten Praxiskennern lernen wollen, dann sind Sie bei SOPHIST an der richtigen Adresse. Ausgestattet mit jahrelanger Erfahrung in unterschiedlichsten Kontexten kümmern wir uns gerne um Sie, Ihre Projekte, Ihre KollegInnen – und das immer bedarfsgerecht, denn Sie stehen im Mittelpunkt unseres Handelns. Wir haben kein Standardvorgehen, sondern passen die Prozesse, Techniken und Methoden individuell an Ihre Bedürfnisse an. Damit können wir gemeinsam mit Ihnen das für Sie optimale Vorgehen entwickeln. Für uns ist – wie schon die alten griechischen Sophisten sagten – der Mensch das Maß aller Dinge.

Wir freuen uns auf Kontakt zu Ihnen.

Ihre SOPHISTen

# Quellenverzeichnis

[Babok®v3] International Institute of Business analysis: BABOK®v3: Leitfaden zum Business Analysis Body of Knowledge, 3. Auflage, Wettenberg 2017

[Bandler75] Bandler, R.; Grinder, J.: The Structure of Magic II. Science and Behaviour Books, Palo Alto/CA, 1975.

[Bandler94] Bandler, R., Grinder, J.: Metasprache und Psychotherapie, Die Struktur der Magie I. 8. Auflage. Junfermann, Paderborn, 1994.

[CPRE20] Pohl, K.; Rupp, C.: Basiswissen Requirements Engineering. Aus- und Weiterbildung zum Certified Professional for Requirements Engineering – Foundation Level nach IREB-Standard, 4. Auflage, Heidelberg, 2020.

[Goodwin09]: Goodwin, K.: Designing For The Digital Age. 1. Auflage, Indianapolis, 2009.

[INCOSE20] INCOSE: Guide to the Systems Engineering Body of Knowledge. Introduction to Systems Engineering (2019), https://www.sebokwiki.org/wiki/Introduction_to_Systems_Engineering (Stand: 09.01.2020)

[ISO26262] ISO-Norm 26262: Road vehicles. Functional safety, 2. Auflage, 2018.

[Pousttchi17] Pousttchi, K.; GITO: Digitale Transformation (2017), http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/technologien-methoden/Informatik--Grundlagen/digitalisierung/digitale-transformation (Stand: 01.04.2020)

[Rupp20] Rupp, Chris: Requirements-Engineering und –Management, Aus der Praxis von klassisch bis agil. 7. Auflage. Hanser, München, 2020.

[SAFe] Scaled Agile Framework: Scaled Agile Framework (o. J.), https://www.scaledagileframework.com/ (Stand: 01.04.2020)

[Wake03] Wake, B.: INVEST in Good Stories, and SMART Tasks (2003), https://xp123.com/articles/invest-in-good-stories-and-smart-tasks (Stand: 25.07.2019)

[Werdich12] Martin Werdich: FMEA – Einführung und Moderation. 2. Auflage. Vieweg & Teubner 2012,

[Wolfe10] Wolfe, P.: Brain Matters Translating Research into Classroom Practice. 2. Auflage. Alexandria, 2010.

# SOPHIST Eigenproduktionen

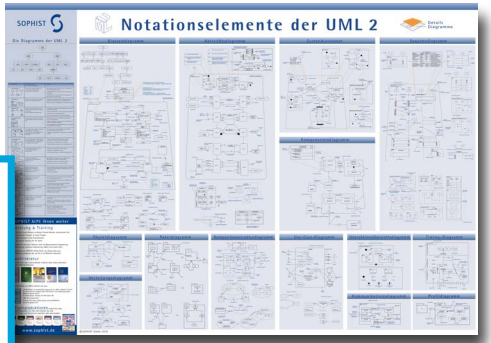
## Wissensträger mal anders!

## Kostenfrei!

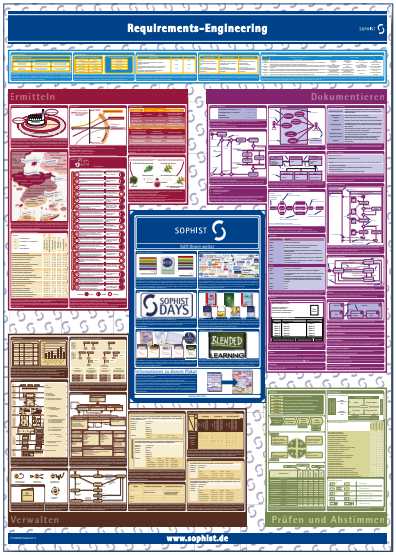
WISSEN for free

## Poster/Plakate

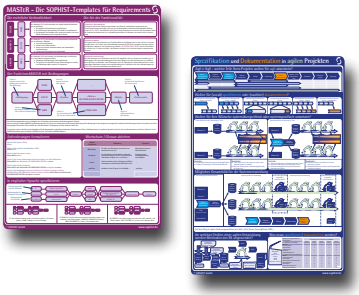
### Das SOPHIST UML-Plakat




### Das SOPHIST RE-Plakat ...




### ... und seine „Kerne“



# Online/Remote Trainings

## Ihre SOPHIST Weiterbildung - an nahezu jedem Ort der Welt

SOPHIST Online/Remote Trainings sind speziell für die Vermittlung von Wissen und Können über das Internet entwickelt worden. Die besondere und sorgfältig durchdachte Ausarbeitung – inhaltlich und didaktisch – sowie eine Teilnehmerzahl von maximal 12 Personen gewährleistet einen perfekten Online-Wissenstransfer.

Für **Einzelpersonen** und **kleinere Teams** eignen sich unsere „Offenen Trainings“ perfekt. Und durch den modularen Aufbau, der Kombination verschiedener Schwerpunkte und der Möglichkeit der individuellen Anpassung, eignen sich Remote/Online Trainings auch perfekt für firmeninterne Weiterbildungen **kompletter Teams.**

Natürlich gibt es auch unsere bekannten CPRE-Zertifizierungstrainings in diesem Format.

## ... egal wo

WISSEN for free

# Broschüren


**www.sophist.de/wissen-for-free**

# SOPHIST

## Kompetenz und Fachwissen

*par excellence*

**Methodenerfinder**

**Speaker**

**Buchautoren**

# Berater Coach Trainer

## Das bieten wir Ihnen an:

Wir unterstützen Sie kompetent, tatkräftig und zielführend sowohl bei der Anpassung Ihrer Entwicklungsprozesse und -methoden als auch bei der Durchführung Ihres Projekts.

Zu unseren Kunden gehören viele weltbekannte Unternehmen. Die Vielzahl an positiven Meinungen und Projektberichten unserer zufriedenen Kunden spricht für sich. Werfen Sie doch einen Blick auf **www.sophist.de/referenzen**

## Unsere Leistungen:

- Verbesserungspotenziale unter Berücksichtigung der Randbedingungen in Ihrer Organisation identifizieren, ausschöpfen und einführen
- Anforderungen und Architekturen angemessen erheben, analysieren sowie vermitteln und dokumentieren
- Unterwegs in einfachen Software-Anwendungen bis hin zu komplexen Systemen
- agil und angepasst zu arbeiten

All das und noch viele weitere Themen aus der Welt des Requirements- und Systems-Engineerings bieten wir Ihnen in Form

# Wie können wir Ihnen helfen?

Gerne arbeiten wir mit Ihnen ein Konzept aus, um Sie bestmöglich in Ihrem Vorhaben zu unterstützen.

**Kontaktieren Sie uns unverbindlich:**

**+49 (0) 911 40 900 - 0**

**heureka@sophist.de**

SOPHIST Trainings

UPDATE

# Was ist daran neu?

Alle Trainings von SOPHIST sind nach neuestem Stand der Wissensvermittlung konzipiert. Unsere Trainer folgen unter anderem den Grundsätzen der Methode „... from the Back of the Room“, um Trainingsinhalte nachhaltig zu vermitteln.

Eine aktivierende Lernumgebung – mehr Bewegung, weniger Text, mehr Interaktion mit den Teilnehmern und überraschende Übungskonzepte – sorgt für Spaß und Effizienz beim Lernen.

# Ihre Vorteile?

Sie profitieren nicht nur von dem Know-How der Methodenführer, sondern auch von einer didaktischen Umsetzung, die ihre Spuren hinterlässt.

# Neugierig geworden?

**Kontaktieren Sie uns unverbindlich:**

**+49 (0) 911 40 900 - 0**

**heureka@sophist.de**

## Grundlagen des Requirements-Engineerings

Bei der Entwicklung eines Systems – sei es nun eine Software, ein eingebettetes System oder ein ganzes Gebäude – ist die Voraussetzung für den Erfolg, dass allen daran Beteiligten bekannt ist, was eigentlich entwickelt werden soll. Das fängt bei den groben Zielen an und endet bei mitunter sehr detaillierten Vorgaben für die Umsetzung.

Die Arbeit eines Requirements-Engineers befasst sich genau mit diesem Thema. Seine Aufgaben sind

- das Ermitteln,
- das Analysieren, das Prüfen und Abstimmen,
- das Vermitteln und/oder Dokumentieren, sowie
- das Verwalten

von Anforderungen an ein System.

Wir möchten Ihnen im Rahmen dieser Broschüre einen Überblick darüber geben, welche Tätigkeiten zu diesen Aufgabenbereichen gehören, und mit was wir, die SOPHISTen, uns seit mittlerweile mehr als 20 Jahren beschäftigen.