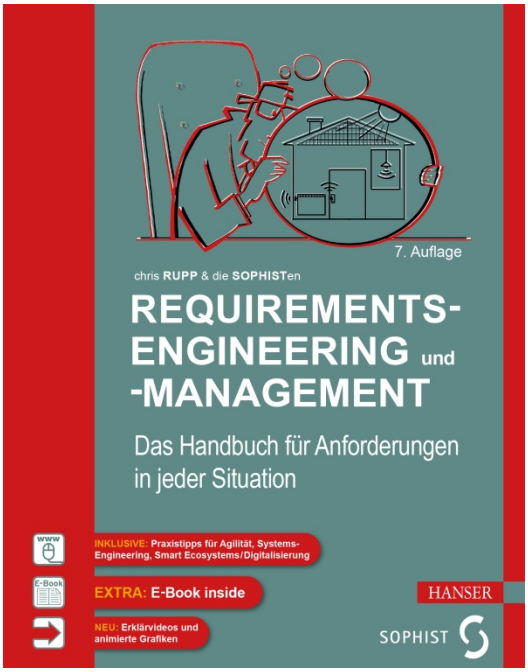
HANSER



Leseprobe

zu

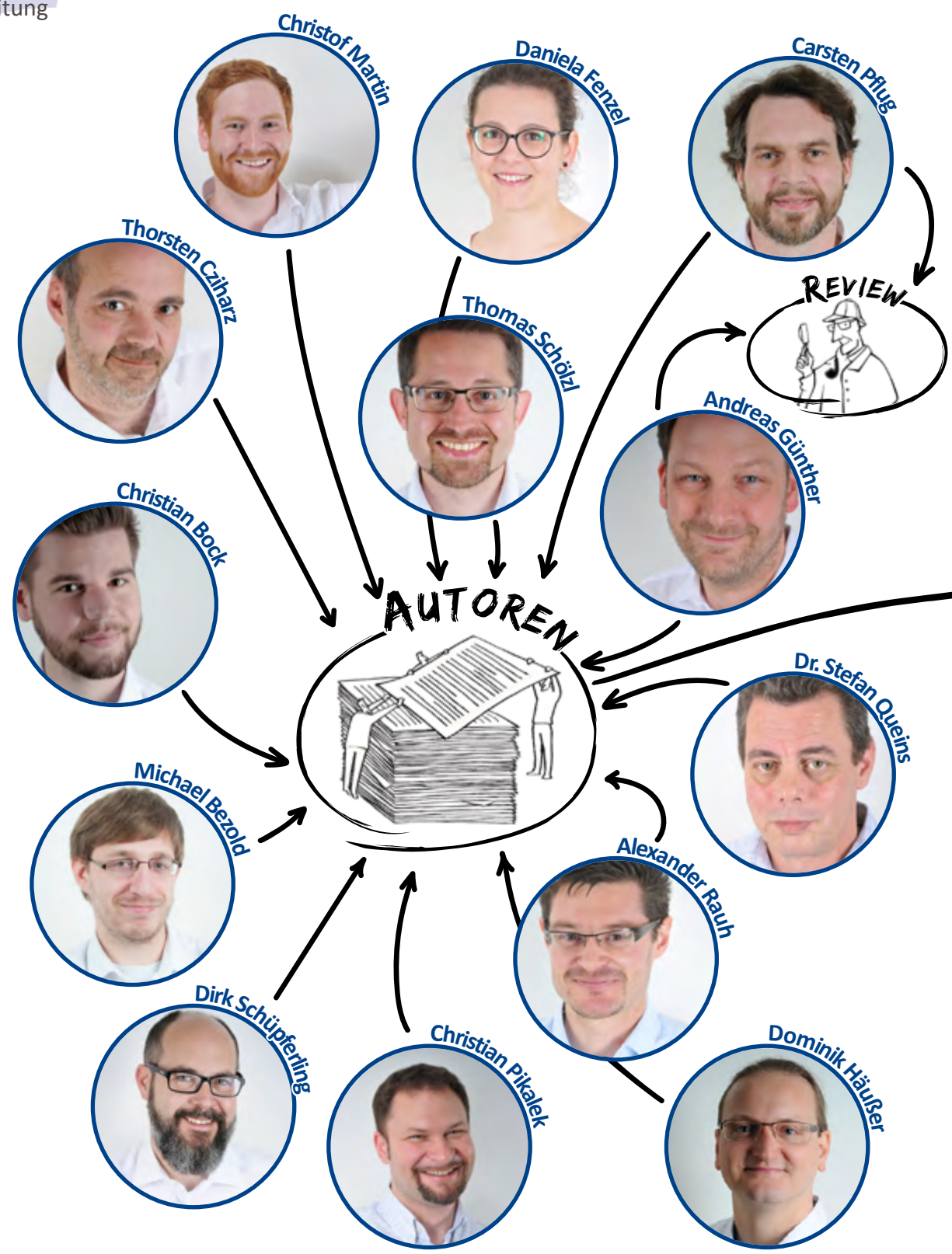
Requirements-Engineering und -Management

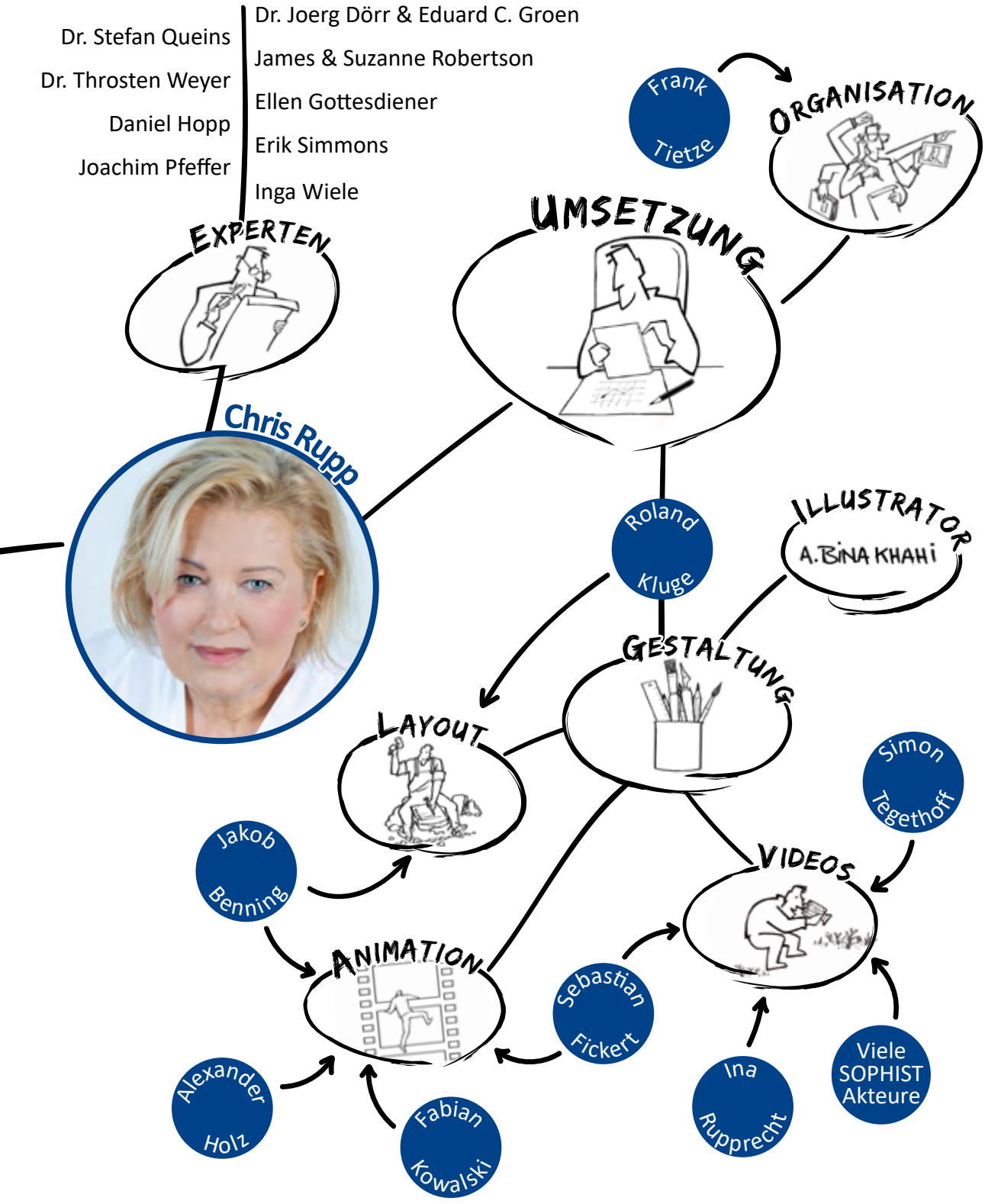
von Chris Rupp & die SOPHISTen

Print-ISBN: 978-3-446-45587-0
E-Book-ISBN: 978-3-446-46430-8

Weitere Informationen und Bestellungen unter
http://www.hanser-fachbuch.de/978-3-446-45587-0
sowie im Buchhandel

Inhaltsverzeichnis

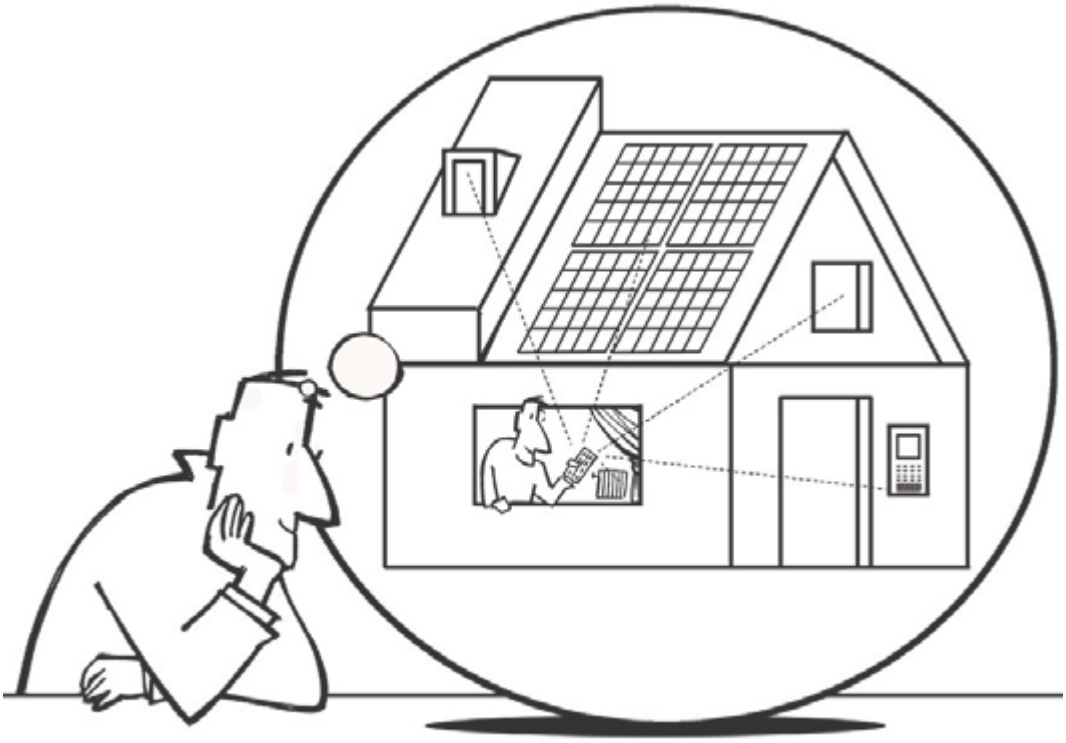

Christof Martin
Daniela Fenzel
Carsten Pflug
Thorsten Cziharz
Thomas Schölzl
REVIEW
Andreas Günther
Christian Bock
AUTOREN
Dr. Stefan Queins
Michael Bezold
Alexander Rauh
Dirk Schüpferling
Christian Pikalek
Dominik Häußer


Dr. Stefan Queins
Dr. Throsten Weyer
Daniel Hopp
Joachim Pfeffer
Dr. Joerg Dörr & Eduard C. Groen
James & Suzanne Robertson
Ellen Gottesdiener
Erik Simmons
Inga Wiele
EXPERTEN
Frank Tietze
ORGANISATION
UMSETZUNG
Chris Rupp
Roland Kluge
ILLUSTRATOR
A.BINA KHAHI
GESTALTUNG
LAYOUT
Simon Tegethoff
Jakob Benning
VIDEOS
ANIMATION
Sebastian Fickert
Alexander Holz
Fabian Kowalski
Ina Rupprecht
Viele SOPHIST Akteure

Stefan Queins, Alexander Rauh

12

Anforderungen analysieren – vom Wunsch zur Absicht



12.1 Überblick über die Analyse von Anforderungen

Wie im vorigen Kapitel angesprochen, ist das Ziel der Analyse von Ursprungsanforderungen (also Anforderungen, die als Ausgangsbasis dienen), belastbare Anforderungen an den gewählten Betrachtungsgegenstand zu ermitteln. Diese Analyse soll angemessen durchgeführt werden, das heißt, es soll nicht mehr Aufwand in diesen Schritt investiert werden, als unbedingt notwendig ist. Leider zeigt der Projektalltag, dass auch dafür nicht immer die benötigten Ressourcen zur Verfügung stehen. Als Ausweg müssen die zur Verfügung stehenden Ressourcen möglichst gewinnbringend eingesetzt werden. Dazu sollten Sie sich zunächst Gedanken darüber machen, welche Qualitätskriterien (siehe Kapitel 3 „Requirements-Engineering im Überblick“) für Ihre Anforderungen im Vordergrund stehen sollten. In den wenigstens Fällen können Sie Anforderungen erzeugen, die zu 100 % allen Qualitätskriterien genügen. Des Weiteren sollten Sie sich immer wieder fragen, mit welchen Anforderungen Sie noch nicht zufrieden sind bzw. an welchen Stellen Ihrer Anforderungssammlung Sie als Nächstes arbeiten müssen, um die Konsumenten der Anforderungen zufriedenzustellen, damit diese die ihnen gestellten Aufgaben erledigen können.

Um diese beiden Gedanken zu unterstützen, wollen wir die Analyse von Anforderungen etwas formalisieren, indem wir in diesem Kapitel sechs unterschiedliche Analyseaufgaben angeben. Mit deren Hilfe könnten Sie eine Anforderungssammlung erzeugen, die zu 100 % den Qualitätskriterien genügt. Oder die – was in der Praxis eher benötigt wird – Ihnen erlauben zu entscheiden, an welcher Stelle Sie Ihre Spezifikation als Nächstes weiter verbessern sollten, sofern Sie noch Ressourcen zur Verfügung haben.

Dabei gilt es zu beachten: Auch wenn durch die Analysetätigkeiten neue Anforderungen entstehen, bedeutet dies nicht, dass Sie diese neuen Anforderungen auch dokumentieren müssen. Sie dienen primär als Ausgangspunkt für weitere Analyseschritte; die Entscheidung, ob Sie diese Anforderungen auch dokumentieren oder beispielsweise nur verbal weitergeben, unterliegt anderen Kriterien (siehe Kapitel 16 „Wegweiser: Anforderungen dokumentieren und vermitteln“).

Durch die Analysetätigkeiten entsteht zunächst im Kopf ein Bild der Anforderungen. Dieses Bild kann, muss aber nicht, vollständig in einer Anforderungsspezifikation oder einem Backlog einer agilen Entwicklung dokumentiert werden.

Auch wenn wir im Folgenden von Anforderungen oder Spezifikationen reden, so sind die Aussagen in diesem Kapitel nicht auf die klassischen Artefakte beschränkt. Die Artefakte einer agilen Entwicklung (Backlog-Items, User-Storys, Akzeptanzkriterien etc., siehe Kapitel 17 „Storytelling, User-Storys und Co.“) können ebenfalls durch die hier beschriebenen Aufgaben entstehen und analysiert werden. Das prinzipielle Vorgehen bei der Analyse von Anforderungen ist in Abbildung 12.1 mit Beispielen für Übergänge zwischen Artefakttypen definiert.

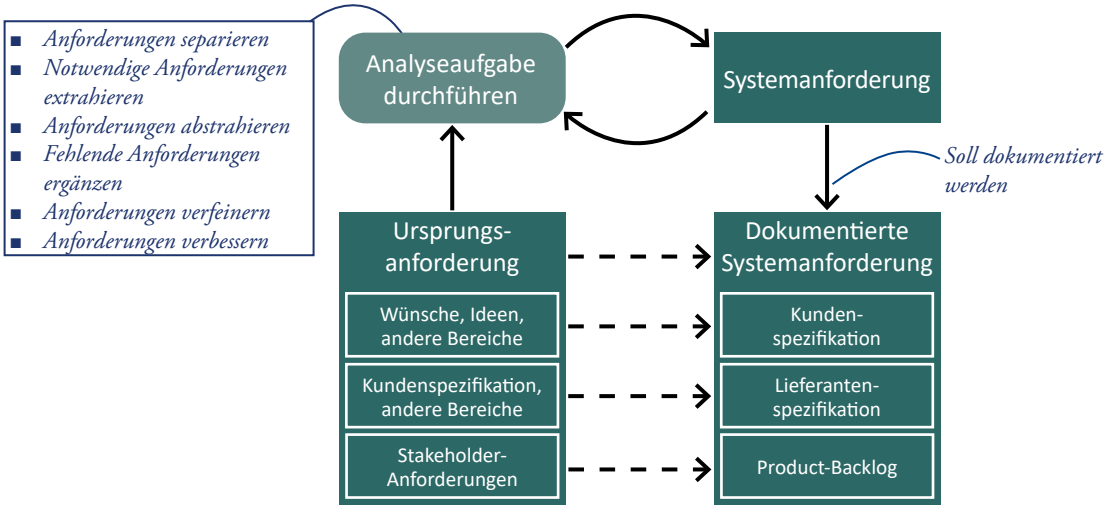


Abbildung 12.1: Vorgehen bei der Analyse von Anforderungen

Wir führen für eine Ursprungsanforderung eine der Analyseaufgaben durch und erhalten dadurch eine Systemanforderung. Unabhängig davon, ob Sie diese Anforderung in die dokumentierten Anforderungen aufnehmen möchten oder nicht, muss die neue Anforderung noch nicht Ihren Ansprüchen genügen. Somit können Sie für diese neu gefundene Systemanforderung eine weitere Analyseaufgabe durchführen, um sie zu verbessern oder daraus neue Anforderungen herzuleiten.

Bevor nun die einzelnen Aufgaben im Detail vorgestellt werden, müssen noch zwei übergreifende Themen geklärt werden. Zunächst möchten wir Ihnen das Gesamtbild der Anforderungen vorstellen, das mithilfe der hier beschriebenen Tätigkeiten ermittelt wird. Danach werden wir die einzelnen Tätigkeiten in Zusammenhang stellen, um dieses Gesamtbild zu erzeugen.

12.1.1 Den Wald trotz vieler Bäume sehen

Mithilfe der hier vorgestellten Aufgaben werden Sie unter anderem viele, eventuell bislang fehlende Anforderungen erzeugen. Dabei gehen wir davon aus, dass jede Anforderung eine andere Anforderung verfeinert (siehe auch Kapitel 3 „Requirements-Engineering im Überblick“). Eine Ausnahme bildet dabei die abstrakteste Ebene von Anforderungen.

Deren 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.

An dieser Stelle kommt die Unterscheidung zwischen nicht-funktionalen und funktionalen Anforderungen mit ins Spiel. Die entstehenden Verfeinerungshierarchien können in diese beiden Arten unterschieden werden, wobei Abhängigkeiten zwischen diesen beiden Teilen bestehen. So kann eine nicht-funktionale Anforderung durch eine funktionale Anforderung verfeinert werden bzw. kann umgekehrt eine nicht-funktionale Anforderung als Begründung für eine Funktion des Systems dienen (siehe Abschnitt 12.2.5 „Anforderungen verfeinern“). In Abbildung 12.2 ist ein unvollständiges Beispiel für diese Zusammenhänge am Beispiel einer Use-Case-Verfeinerung gegeben.

Allerdings kann eine Anforderung auch aus zwei abstrakteren Anforderungen resultieren. Damit müsste man anstatt von Bäumen von Graphen reden, da in diesem Fall ein Knoten in dem Baum mehr als ein Elternelement besitzt. In dem Beispiel in Abbildung 12.2 könnte dies die Forderung nach der Erkennung der Bewegung sein, die in anderen Use-Case-Abläufen zusätzlich benötigt werden könnte. Für die Vorstellung der Analyseaufgaben in diesem Kapitel wollen wir jedoch diesen Fall außer Acht lassen und weiterhin in Bäumen denken. Wie Sie in Kapitel 21 „Strukturen und Zustände“ lesen werden, kann die Verfeinerungshierarchie zur Strukturierung einer Anforderungsspezifikation genutzt werden. Dabei muss der Fall mehrerer abstrakter Anforderungen als Begründung zu einer Anforderung gesondert betrachtet werden.

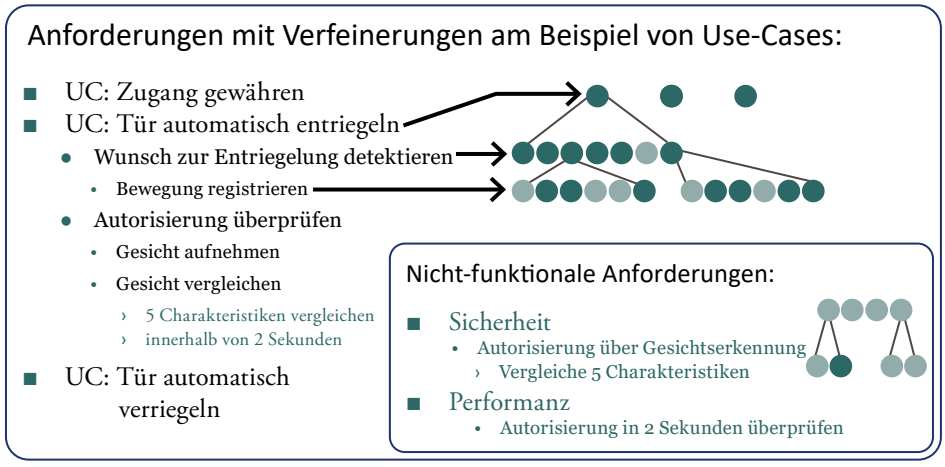


Abbildung 12.2: Zusammenhang zwischen Anforderungen

12.1.2 Der Ablauf bei der Anforderungsanalyse

In diesem Abschnitt möchten wir die Aufgaben im Überblick darstellen und sie in eine Reihenfolge bringen. Dies soll nur als Leseanleitung dienen, da die Aufgaben nicht

unbedingt in dieser Reihenfolge bearbeitet werden müssen. Manchmal kann eine Aufgabe für eine betrachtete Anforderung ausgelassen werden, manchmal muss sie zweimal durchgeführt werden. Wann es sinnvoll ist, eine der genannten Aufgaben auch wirklich durchzuführen, lesen Sie bitte in den folgenden Abschnitten.

Als Eingabe in die Analyse nehmen wir eine Anforderung an, die Ihnen von einem Stakeholder mitgeteilt wurde oder im Rahmen eines Lastenhefts bei Ihnen aufgeschlagen ist. In dem Schritt *Anforderungen separieren* sollten Sie als Erstes diese Ursprungsanforderung gegebenenfalls in mehrere Anforderungen zerlegen, um diese in den nächsten Schritten getrennt voneinander betrachten zu können. Im zweiten Schritt (*Notwendige Anforderungen extrahieren*) sollten 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 Schritte sollten Sie auf alle Ursprungsanforderungen anwenden, um so einen guten Startpunkt für die weitere Analyse zu erzeugen.

Die nächste Aufgabe ist es, zu einer Anforderung abstraktere Anforderungen zu finden (*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 (*Anforderungen verfeinern*), in der Sie nun entscheiden können, welche der bestehenden Knoten in den Bäumen weiter verfeinert werden sollen.

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

In der Praxis werden Sie bei der Anwendung 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.

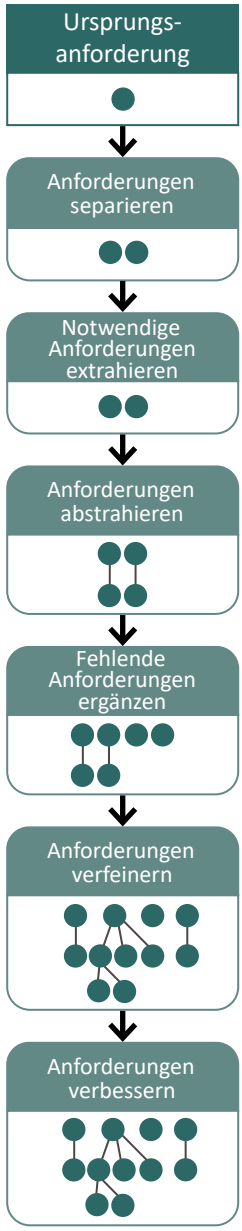


Abbildung 12.3:
Die Analyseaufgaben in der Übersicht

www.sophist.de/re7p/k12v1/

GIF www.sophist.de/re7p/k12a3/

12.2 Die Aufgaben im Detail

Kommen wir nun zum Kern dieses Kapitels: Die Beschreibung der einzelnen Aufgaben. Sie werden in einer bestimmten Reihenfolge dargestellt, jedoch können die Schritte wiederholt oder in einer anderen Reihenfolge für eine Ursprungsanforderung durchgeführt werden.

Für die Beschreibung gehen wir von vier Ursprungsanforderungen an unser Smart-Home-System (SHS) aus, wie die Stakeholder sie Ihnen genannt haben könnten.

Das SHS muss die Erlaubnis zum Öffnen der Tür überprüfen und innerhalb von 2 Sekunden die Tür öffnen. (U1)

Das SHS muss jede Entriegelung einer Tür auf konfigurierten Handys und Tablets innerhalb von 2 Sekunden anzeigen. (U2)

Der Türsensor muss den Status einer Tür (offen/geschlossen) melden. (U3)

Die Nutzung des SHS muss so einfach wie möglich sein. (U4)

Durch die Bearbeitung der Ursprungsanforderungen gemäß der sechs Aufgaben werden sie sukzessive in belastbare Systemanforderungen, in die Ergebnisanforderungen, überführt. Dabei werden sie sowohl um neue Anforderungen ergänzt als auch in Richtung der Eindeutigkeit und Notwendigkeit verbessert. Hier kommen auch die in Kapitel 9 „Das SOPHIST-*RE*gelwerk“ vorgestellten Regeln ins Spiel, die Ihnen Hinweise geben, welche Defizite in einer betrachteten Anforderung existieren können.

12.2.1 Anforderungen separieren

Ihre erste Aufgabe bei der Analyse von vorgegebenen Anforderungen ist es, sie in ihre Bestandteile zu zerlegen. Betrachten wir dazu die folgende Beispielanforderung.

Das SHS muss die Erlaubnis zum Öffnen der Tür überprüfen und innerhalb von 2 Sekunden die Tür öffnen. (U1)

Es ist offensichtlich, dass in dieser Anforderung zwei Funktionen von dem System gefordert werden: Das *SHS soll die Erlaubnis überprüfen* und die *Tür öffnen*. Dabei ist nur implizit ausgedrückt, dass es die Tür natürlich nur bei erfolgreicher Prüfung öffnen soll. Daraus folgt ein Grund, warum wir in einer Anforderung nur eine Funktion (bzw. eine funktionale Anforderung) fordern wollen: Der Zusammenhang zwischen den Funktionen kann explizit ausgedrückt werden.

Die beiden Funktionen in der obigen Anforderung und die (nicht eindeutig dem *Erlaubnis überprüfen* oder dem *Tür öffnen* zuordenbare) Anforderung an die Performanz bringen uns zu einem weiteren Grund für die Zerlegung: Die nachfolgende Bearbeitung der Anforderungen kann getrennt voneinander erfolgen. Für die nachfolgenden Entwicklungsschritte kann eine getrennte Betrachtung sinnvoll sein. Hierzu einige Beispiele:

- Die Funktionen sollen getrennt voneinander getestet werden: Dabei würde die gesamte Ursprungsanforderung auch bei einem „Fail“ von einem Teil der Anforderung als „nicht bestanden“ gewertet werden müssen.

- Die Anforderung an die Performanz soll später entwickelt werden: Auch hier hätten Sie nur die Chance, die gesamte Anforderung als „realisiert“ zu bewerten, und nicht den tatsächlich entwickelten Teil. In einer agilen Entwicklung spricht man hier von einem Zerlegen einer User-Story.
- Nur ein Teil der Anforderung soll weiter verfeinert werden: Es sollte beim Lesen der Anforderung offensichtlich sein, welchen Teil der Anforderung Sie mit weiteren Anforderungen weiter beschreiben.

Wir können uns also die folgenden Fragen stellen, um zu entscheiden, ob eine Anforderung zerlegt werden sollte oder nicht.

- Beinhaltet die Anforderung unterschiedliche Funktionen des Systems?
- Existieren unterschiedliche Bedingungen für die einzelnen Anteile der Anforderung?
- Beinhaltet die Anforderung eine Qualitätsanforderung zu einer funktionalen Anforderung?

Sobald Sie eine oder mehrere der Fragen mit Ja beantworten, haben Sie gute Gründe, Ihre Anforderungen in ihre Bestandteile zu zerlegen und diese im Weiteren als eigenständige Anforderungen zu betrachten. Falls Sie sich unsicher sind, sollten Sie diese Separierung trotzdem durchführen, da im weiteren Verlauf der Entwicklung eher Gründe für eine getrennte Betrachtung hinzukommen werden als umgekehrt.

Für die Ursprungsanforderung U1 erhalten Sie so die drei Ergebnisanforderungen (E1 bis E3):

Das SHS muss die Erlaubnis zum Öffnen einer Tür überprüfen und innerhalb von 2 Sekunden die Tür öffnen. (U1)

Das SHS muss die Erlaubnis zum Öffnen einer Tür überprüfen. (E1)

Sobald das SHS die Erlaubnis zum Öffnen einer Tür überprüft hat und falls die Überprüfung positiv war, muss das SHS die entsprechende Tür öffnen. (E2)

Die Funktion Tür öffnen muss eine Ausführungszeit von maximal 2 Sekunden aufweisen. (E3)

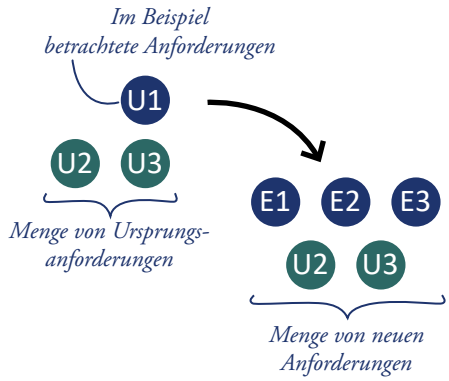
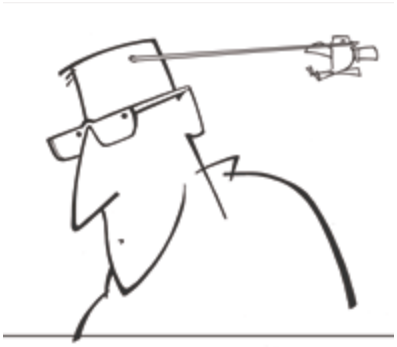


Abbildung 12.4: Ergebnis der Aufgabe *Anforderungen separieren*

Die Umformulierung beinhaltet einige Annahmen, die nicht explizit in der Ursprungsanforderung enthalten waren, z. B. die Zuordnung der zwei Sekunden zum Öffnen der Tür nach Überprüfung. Seien Sie vorsichtig bei solchen Annahmen und lassen Sie sich diese von Ihrem Stakeholder bestätigen.

Abbildung 12.4 zeigt die Ausgangsmenge und die Menge der durch die Separierung erzeugten Anforderungen. Der hier dargestellte, noch sehr kleine „Wald“ auf der rechten Seite wird durch die folgenden Aufgaben in der Analyse sukzessive ausgebaut.

12.2.2 Notwendige Anforderungen extrahieren



Häufig drücken die Anforderungen der Stakeholder nur Wünsche aus, die zum einen nicht alle erfüllt werden können und die zum anderen vielleicht nicht alleine von dem betrachteten System realisiert werden können.

Der wohl wichtigste Schritt in der Analyse von Anforderungen ist es demnach, diejenigen Anforderungen aus den Ursprungsanforderungen zu extrahieren, die tatsächliche Anforderungen an das System darstellen. Erst dann haben Sie nicht mehr und nicht weniger von Ihrem System gefordert, als es leisten soll. Bei dieser Tätigkeit unterscheiden wir zwei Fälle.

1. Fall: Das System realisiert nur einen Teil der Anforderung

Das wohl häufigste Defizit in den Ursprungsanforderungen besteht darin, dass dort mehr verlangt wird, als das betrachtete System wirklich leisten kann bzw. leisten soll. In der Praxis findet man immer wieder zwei Gründe dafür.

Entweder wünscht sich der Stakeholder etwas von dem neuen System, das von dem System mit den gegebenen Randbedingungen (Entwicklungszeit und -kosten bzw. Herstellungskosten des Systems) nicht realisiert werden kann. Oder der Wunsch muss in einer Zusammenarbeit zwischen dem betrachteten System und seinen Nachbarsystemen (z. B. von anderen Lieferanten verantwortet) realisiert werden. Dies deutet darauf hin, dass der Architekturschritt eine Ebene oberhalb zur Abgrenzung des Systemkontextes nicht vollständig durchgeführt oder bei der Formulierung der Anforderungen nicht korrekt berücksichtigt wurde (siehe auch Kapitel 3 „Requirements-Engineering im Überblick“). Stellen Sie sich also die Frage:

- Welchen Teil der geforderten Funktionalität soll das System realisieren?

Betrachten wir die Konsequenzen der Anforderung E3 an die Performanz in der Realisierung, so stellt sich heraus, dass dafür leistungsfähige Motoren existieren müssen, die Ihrem Kunden vielleicht zu teuer sind. Eine Alternative wäre, nur das Entriegeln der Tür und nicht das zusätzliche Öffnen der Tür zu unterstützen. Diese Änderung muss dann auch in den bereits bestehenden Anforderungen nachvollzogen werden, wodurch zuvor gefundene Anforderungen durch neue Versionen ersetzt werden.

Sobald das SHS die Erlaubnis zum Öffnen einer Tür überprüft hat und falls die Überprüfung positiv war, muss das SHS die entsprechende Tür öffnen. (E2)

Das SHS muss die Erlaubnis zum Entriegeln einer Tür überprüfen. (E1‘)

Sobald das SHS die Erlaubnis zum Entriegeln einer Tür überprüft hat und falls die Überprüfung positiv war, muss das SHS die entsprechende Tür entriegeln. (E2‘)

Die Funktion Tür entriegeln muss eine Ausführungszeit von maximal 2 Sekunden aufweisen. (E3‘)

Das nächste Beispiel beschäftigt sich mit der Einbeziehung von Nachbarsystemen. Hier gehen wir von der Anforderung U2 aus. Da das angesprochene Handy oder Tablet jedoch nicht Bestandteil des SHS sein soll (siehe Kapitel 6 „Ziele, Informanten und Fesseln“), folgen als tatsächliche Funktionen des Systems leicht geänderte Anforderungen. Dabei wurde auch für die implizite Performanzanforderung in U2 die Anforderung E5 definiert, um den geänderten Funktionalitäten gerecht zu werden.

Das SHS muss jede Entriegelung einer Tür auf konfigurierten Handys und Tablets innerhalb von 2 Sekunden anzeigen. (U2)

Sobald das SHS eine Tür entriegelt hat, muss das SHS die Entriegelung an die konfigurierten Handys und Tablets senden. (E4)

Das Senden muss eine Ausführungszeit von maximal 1 Sekunde aufweisen. (E5)

2. Fall: Die Anforderung bezieht sich auf eine Komponente

Ebenfalls häufig tritt die Situation auf, dass ein Stakeholder eine Anforderung stellt, die sich nur an eine Komponente des Systems richtet. Man macht sich das Leben leicht und nimmt an, dass es eine solche Komponente gibt, vergisst dabei jedoch, die Begründung für die geforderte Komponentenanforderung als Systemanforderung zu fordern. Ihre Aufgabe ist es, diese Systemanforderung zu formulieren, da jede Anforderung an eine Komponente durch eine Systemanforderung begründet werden sollte. Zusätzlich haben Sie so eine weitere Systemanforderung gefunden, die es wert sein kann, weiter beschrieben zu werden (siehe Abschnitt 12.2.5 „Anforderungen verfeinern“). Beantworten Sie für jede Anforderung an einen Teil des Systems die folgende Frage:

- Wegen welcher Systemanforderung existiert die gegebene Komponentenanforderung?

Auch hier haben wir wieder ein Beispiel. Die Anforderung U3 richtet sich an die Komponente Türsensor des Systems, woraus aber eine neue Anforderung an das System abgeleitet werden kann.

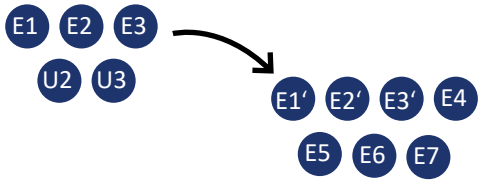


Abbildung 12.5: Ergebnis der Aufgabe *Notwendige Anforderungen extrahieren*

Der Türsensor muss den Status einer Tür (offen/geschlossen) melden. (U3)

Das SHS muss das Öffnen einer Tür erkennen. (E6)

Das SHS muss das Schließen einer Tür erkennen. (E7)

Die Aufgabe, die richtigen Anforderungen herauszufinden, ist auch in der agilen Entwicklung von entscheidender Bedeutung. Beim Formulieren der User-Storys sollte der Product-Owner sich schon auf das für das System Relevante konzentrieren. Dies erspart Nachfragen und Änderungen während der Refinement-Besprechungen mit dem Entwicklungsteam.

12.2.3 Anforderungen abstrahieren

Mit dieser Aufgabe werden zu den Anforderungen, die bereits gefunden wurden, abstraktere Anforderungen definiert. Diese bilden den Ausgangspunkt für die nachfolgende Verfeinerung (siehe Abschnitt 12.2.5 „Anforderungen verfeinern“). Erst durch diesen Schritt der Abstraktion erhalten Sie eine vollständige, wenn auch grobe, Sicht auf die Anforderungen an Ihr System. Des Weiteren werden die bisher gefundenen Anforderungen dadurch in Zusammenhang gesetzt, dass die abstrakten Anforderungen eine gemeinsame Basis für sie bilden.

Für jede Anforderung können Sie sich die folgenden Fragen stellen:

- In welchem Ablauf soll das System die geforderte Funktionalität durchführen bzw. außerhalb des Systems zur Verfügung stellen?
- Bezieht sich eine geforderte Eigenschaft des Systems auf eine Funktion des Systems?
- Zu welchem Problem wird mit der geforderten Funktionalität oder Eigenschaft eine Lösung vorgegeben?

Die Antwort auf die erste Frage wird Sie zu einer neuen, abstrakteren Funktion des Systems bringen, während eine Antwort auf die zweite Frage zu einer Funktion auf einer beliebigen Abstraktionsebene führt. Eine Antwort auf die dritte Frage führt Sie zu einer nicht-funktionalen Anforderung, einer Qualitätsanforderung.

Für eine neu gefundene Anforderung können Sie nun diesen Schritt erneut durchführen, um auf eine noch abstraktere Anforderung zu kommen. Führen Sie diese Abstraktionen so oft durch, bis Sie:

- einen Use-Case erhalten haben oder
- eine der in Kapitel 13 „ Nicht-funktionale Anforderungen“ vorgestellten Kategorien von nicht-funktionalen Anforderungen erreicht haben.

Betrachten wir die mehrfache Durchführung dieser Abstraktion an einem Beispiel anhand der Anforderungen, die wir in den vorherigen Aufgaben gefunden haben:

Sobald das SHS die Erlaubnis zum Entriegeln einer Tür überprüft hat und falls die Überprüfung positiv war, muss das SHS die entsprechende Tür entriegeln. (E2‘)

Die Frage, in welchem Ablauf das SHS die Tür entriegeln soll, bringt uns zu folgender Anforderung:

Das SHS muss BewohnerInnen die Möglichkeit bieten, eine Tür ohne manuelle Bedienung zu entriegeln. (UC1)

Dies ist eine abstraktere Anforderung als E2‘, da das SHS mehr Schritte für die Anforderung UC1 durchführen muss, als nur die Tür zu entriegeln. Versucht man nun, die erste Fragestellung auf die neu gefundene Anforderung anzuwenden, so kann man sich viele Abläufe vorstellen, im Rahmen derer die BewohnerInnen das Haus betreten möchten

(sie kommen vom Einkaufen nach Hause, sie haben etwas zu Hause vergessen etc.). An allen diesen Abläufen wird das SHS aber „nur“ mit dem Entriegeln der Tür beteiligt sein, weswegen die gefundene Anforderung UC1 einen Use-Case und damit eine Wurzel für viele Anforderungen an das SHS darstellt.

In einem zweiten Beispiel betrachten wir als Ausgangspunkt wieder wie oben die Anforderung E2‘ und stellen uns nun die Frage, weshalb die entsprechenden Bedingungen dort angegeben sind. Als Antwort können wir die Anforderung E8 erhalten.

Nur Befugte dürfen eine Tür entriegeln. (E8)

Dies ist eine Anforderung, die man sofort als Sicherheitsanforderung identifizieren kann, und dementsprechend unterhalb dieser Kategorie (in der Abbildung 12.6 als K1 bezeichnet) anordnen würde.

Als letztes Beispiel gehen wir von der Anforderung E3‘ aus:

Die Funktion Tür entriegeln muss eine Ausführungszeit von maximal 2 Sekunden aufweisen. (E3‘)

Die Frage nach der Funktion, auf die sich die gegebene Ausführungszeit bezieht, führt uns auf die schon vorher gefundene Funktion des Entriegelns der Tür, die in der Anforderung E2‘ beschrieben wurde. Hier erhalten wir also keine neue Anforderung.

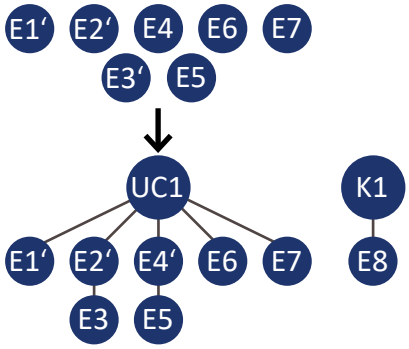


Abbildung 12.6: Ergebnis der Aufgabe *Anforderungen abstrahieren*

Unser Tipp: Führen Sie als Product-Owner in einer agilen Entwicklung diese Aufgabe auf jeden Fall durch. Entweder Sie erreichen dann die Ebene von Epics oder Sie erhalten eine abstraktere User-Story, die als Motivation für die ursprüngliche User-Story dienen kann und die Sie danach in kleinere User-Storys zerlegen können (siehe Abschnitt 12.2.5 „Anforderungen verfeinern“). Durch die Verfeinerung erhalten Sie neue User-Storys, die die ursprüngliche User-Story sinnvoll ergänzen.

12.2.4 Fehlende Anforderungen ergänzen

Mithilfe der zuvor definierten Aufgaben wurden die Anforderungen gefunden, die sich direkt aus den Ursprungsanforderungen ableiten ließen. Diese werden nun um die Aspekte ergänzt, die nicht vom Stakeholder vorgegeben wurden, aber zum Teil implizit in den bisher gefundenen Anforderungen gefordert werden.

Auch hier werden wieder einige Fragen gestellt, die Sie zu den fehlenden Anforderungen führen:

- Mit welcher Funktion wird eine geforderte Eigenschaft eines Objekts oder eine Bedingung einer Anforderung sichergestellt?
- In welcher Funktion wird eine Ausgabe an einer Schnittstelle erzeugt?

- In welcher Funktion wird eine Eingabe an einer Schnittstelle verwendet?
- Welche weiteren Funktionen werden auf oberster Ebene benötigt?
- Welche weiteren Kategorien von nicht-funktionalen Anforderungen sind für Ihr System relevant?

Weitere Indikatoren finden Sie in Kapitel 9 „Das SOPHIST-*RE*gelwerk“.

Während die ersten drei Fragen die impliziten Annahmen in den Anforderungen adressieren, dienen die letzten beiden Fragen zur Ermittlung der Basis- und zum Teil auch der Leistungsfaktoren Ihres Systems (siehe Kapitel 8 „Anforderungsermittlung“).

Auch hier wollen wir wieder ein paar Beispiele darstellen, um die Anwendung der Fragen zu verdeutlichen. Beginnen wir mit der Anforderung E4.

Sobald das SHS eine Tür entriegelt hat, muss das SHS die Entriegelung an die konfigurierten Handys und Tablets senden. (E4)

Wendet man die erste Frage von oben auf diese Anforderung an, so ergibt sich eine neue Anforderung, die aus der gegebenen Bedingung resultiert:

Das SHS muss eine Tür entriegeln.

Diese Anforderung entspricht einer Anforderung, die in ausführlicherer Version schon vorher definiert wurde (E2‘ aus Abschnitt 12.2.2 „Notwendige Anforderungen extrahieren“); Sie muss also nicht erneut aufgenommen werden. Aus der Bedingung in Anforderung E2‘ folgt jedoch die folgende Anforderung:

Das SHS muss die Erlaubnis zum Entriegeln einer Tür überprüfen.

Auch diese Anforderung wurde in einem vorherigen Schritt als E1‘ gefunden und muss dementsprechend ebenfalls nicht als neue Anforderung definiert werden.

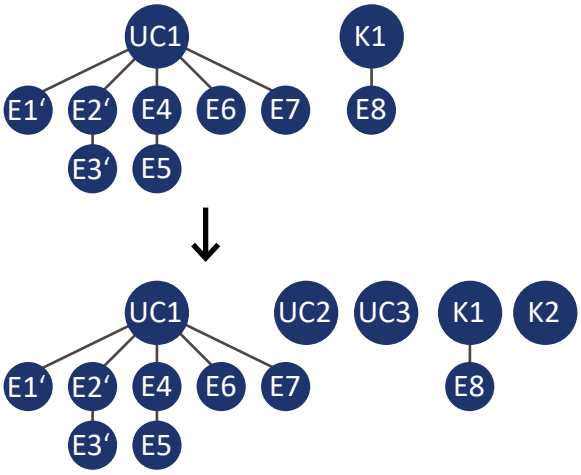


Abbildung 12.7: Ergebnis der Aufgabe *Fehlende Anforderungen ergänzen*

Aus der in E4 erwähnten Konfigurierbarkeit folgt:

Das SHS muss berechtigten Personen die Möglichkeit bieten, Handys und Tablets als Empfangsgerät zu konfigurieren. (UC2)

Bitte beachten Sie, dass bei dieser Anforderung neue Informationen hinzugekommen sind, die natürlich wieder mit Ihrem Stakeholder abgestimmt werden müssen. Sie ist direkt als Use-Case definiert, da eine weitere Abstraktion aus dem Betrachtungsgegenstand des SHS herausführen würde (siehe Abschnitt 12.2.3 „Anforderungen abstrahieren“).

Als letztes Beispiel betrachten wir nun die Use-Cases, um die oberste Ebene von Anforderungen zu vervollständigen.

Aus der Anforderung, dass das SHS die Tür entriegeln soll, ergibt sich sofort die Frage, ob das SHS diese Tür nicht auch verriegeln sollte. Nach Absprache mit dem Stakeholder könnte man folgende Anforderung, wiederum als Use-Case, formulieren:

Das SHS muss eine Tür automatisch verriegeln. (UC3)

Ähnlich verhält es sich mit den Kategorien von nicht-funktionalen Anforderungen. Überprüfen Sie die bisher gefundenen Kategorien, ob nicht eine für Ihr System wichtige Kategorie bisher noch nicht gefunden wurde. Für unser SHS könnte man sich sicher Anforderungen an die Performanz oder Bedienbarkeit vorstellen.

Mit den hier erwähnten Beispielen ergibt sich zusammenfassend das Bild aus Abbildung 12.7 für unsere bisher gefunden Anforderungen. Gegenüber der Abbildung 12.6 wurden lediglich zwei neue Use-Cases und eine neue Kategorie von nicht-funktionalen Anforderungen (K2: Performanzanforderungen) aufgenommen. Eigentlich könnten noch weitere Anforderungen, insbesondere Use-Cases, gefunden werden; wir haben diese aus Gründen der Übersichtlichkeit nicht weiter betrachtet.

Auch diese Aufgabe werden Sie, bewusst oder unbewusst, in einer agilen Entwicklung beim Definieren von User-Storys oder auch bei Refinement-Besprechungen durchführen. Damit identifizieren Sie z. B. neue User-Storys, die dazu dienen, in der ursprünglichen User-Story getroffene Annahmen implementieren zu lassen.

12.2.5 Anforderungen verfeinern

Kommen wir nun zu der Aufgabe mit den meisten Freiheitsgraden in der Analyse der Anforderungen, dem Verfeinern. Hier können Sie für jede bisher gefundene Anforderung (also für jeden Knoten in dem bisher entstandenen Baum) entscheiden, ob Sie diese Anforderung noch genauer beschreiben möchten. Sie sollten diesen Schritt für eine Anforderung in den folgenden Fällen nicht durchführen:

- Sie sind mit jeder fachlichen Lösung für die Anforderung zufrieden.
- Sie sind sich sicher, dass es zum allgemeinen Wissen gehört, was genau unter der Anforderung zu verstehen ist.
- Das Verb, das den in einer funktionalen Anforderung geforderten Prozess beschreibt, ist im Glossar definiert und muss deswegen nicht über eine Verfeinerung genauer beschrieben werden.

Wenn Sie sich nicht zu 100 % sicher sind, dass einer der angegebenen Gründe zutrifft, sollten Sie diese Anforderung verfeinern. Das wird jedoch zu einem sehr hohen Aufwand in der Analyse führen. Diesen Aspekt werden wir in Abschnitt 12.3 „Angemessener Einsatz der Tätigkeiten“ betrachten. Falls Sie die erste Frage zwar verneinen müssen, Sie jedoch mit vielen, aber nicht allen Lösungen einverstanden sind, so können Sie als Verfeinerung einer Anforderung auch Lösungen mithilfe des entsprechenden Modaloperators ausschließen (siehe Kapitel 3 „Requirements-Engineering im Überblick“).

Stellen Sie sich die folgenden Fragen, um eine Anforderung zu verfeinern:

- Wie ist der Ablauf der Interaktion zwischen Personen oder Nachbarsystemen und dem betrachteten System bzw. welche Schritte muss das System für eine geforderte Funktionalität durchführen?
- Wie kann das System eine geforderte nicht-funktionale Eigenschaft sicherstellen?

Für das Finden einer Verfeinerung hat es sich als zielführend herausgestellt, zunächst ungeordnet alle die Anforderungen zu betrachten, die Ihnen bei der zu verfeinernden Anforderung wichtig sind. Bringen Sie die Anforderungen in eine Ordnung (z. B. einen Ablauf) und erweitern Sie diese um fehlende Anforderungen. So erhalten Sie eine vollständige Verfeinerung der eingangs betrachteten Anforderung. In der Praxis hat es sich bewährt, bestimmte Aspekte (z. B. eine Fehlerbehandlung) bei dieser Verfeinerung nicht zu betrachten, sondern gesondert und funktionsübergreifend zu beschreiben.

Als erstes Beispiel betrachten wir den identifizierten Use-Case UC1. Auch wenn schon einige Anforderungen als Verfeinerung gefunden wurden, so wird eine ausführliche Betrachtung weitere Anforderungen liefern. Als Ergebnis könnte ein Ablaufdiagramm wie in Kapitel 18 „Anforderungen modellieren“ entstehen, wobei die dort dargestellten Aktionen als Anforderung interpretiert werden. Für diese Anforderungen kann dann eine erneute Verfeinerung notwendig sein. Betrachten wir hierzu wieder die Anforderung E1‘.

Das SHS muss die Erlaubnis zum Entriegeln einer Tür überprüfen. (E1‘)

Falls Sie sich noch nicht auf eine Lösung festlegen wollen, aber eine Authentifizierung über einen PIN-Code ausschließen wollen, so können Sie formulieren:

Die Authentifizierung zum Entriegeln der Tür darf nicht über die Eingabe eines PIN-Codes gelöst werden. (E9)

Herausfordernder wird die Verfeinerung von nicht-funktionalen Anforderungen. Es ist im Allgemeinen ein iterativer Prozess, alle Anforderungen zu formulieren, die eine sehr abstrakte Anforderung verfeinern. Ein Ansatz ist es, die Auswirkungen einer nicht-funktionalen Anforderung bei der Verfeinerung von funktionalen Anforderungen zu betrachten. Zusätzlich können weitere technologische Anforderungen an das System folgen. Als Beispiel hierzu betrachten wir eine allgemeine Anforderung an die Bedienbarkeit.

Die Nutzung des SHS muss so einfach wie möglich sein. (U4)

Diese Anforderung lässt jede Menge Spielraum, den es festzulegen gilt. Eine Auswirkung dieser Anforderung haben wir schon in E9 angegeben, da wir keine Eingabe eines komplizierten Pin-Codes haben wollen. Darüber hinaus kann man eine allgemeine Anforderung als Vorgabe für die Realisierung formulieren:

Das Ein- und Ausschalten des Lichts muss in jedem Raum über einen Tastschalter realisiert werden. (E10)

In einer agilen Entwicklung entspricht dieser Schritt dem Aufteilen einer User-Story, z. B. ausgelöst durch die Diskussionen mit dem Entwicklungsteam während einer Refinement-Besprechung. Anstelle der einen ursprünglichen User-Story werden zwei oder mehr User-Storys in die Entwicklung übergeben (siehe auch Abschnitt 17.2.1 „Verschiedene Detaillierungsebenen von User-Storys“).

12.2.6 Anforderungen verbessern

Mit den bisher vorgestellten Aufgaben haben wir aus den vorgegebenen Anforderungen eine Vielzahl von Systemanforderungen abgeleitet. In dieser letzten Aufgabe geht es nun darum, jede der zuvor gefundenen Anforderungen zu überprüfen und gegebenenfalls zu verbessern, um die Qualität jeder einzelnen Anforderung sicherzustellen.

Betrachten Sie jede einzelne Anforderung und stellen Sie sich die folgenden Fragen:

- Ist es eindeutig, wann ein Test einer Anforderung positiv oder negativ bewertet wird?
- Sind die in einer Anforderung verwendeten Begriffe durchgängig verwendet und in einem Glossar definiert?
- Existieren überflüssige Anteile in der Anforderung?
- Sind die Bedingungen in einer Anforderung vollständig und korrekt?

Gerade die letzte Frage führt zu einer Änderung in einer Anforderung, wodurch es notwendig sein wird, die zuvor vorgestellten Aufgaben für diese Anforderung erneut durchzuführen. Betrachten wir hierzu als Beispiel eine zuvor ermittelte Anforderung.

Sobald das SHS die Erlaubnis zum Entriegeln einer Tür überprüft hat und falls die Überprüfung positiv war, muss das SHS die entsprechende Tür entriegeln. (E2‘)

Diese erscheint zunächst vollständig, jedoch sollte man als Bedingung mit aufnehmen, dass die Tür auch verriegelt ist, sodass die Entriegelung nicht zu einem Fehler führt. Es folgt die verbesserte Anforderung E2“:

Sobald das SHS die Erlaubnis zum Entriegeln einer Tür überprüft hat und falls die Überprüfung positiv war und falls die Tür verriegelt ist, muss das SHS die entsprechende Tür entriegeln. (E2“)

Für den neu hinzugefügten Teil in der Anforderung sollten Sie nun wieder die Aufgabe *Fehlende Anforderungen ergänzen* durchführen. Nur damit stellen Sie sicher, dass das SHS auch den Verriegelungsstatus überprüft.

Sowohl die Frage nach der Korrektheit der Bedingung als auch die Frage nach der eindeutigen Verwendung von Begriffen bringt uns dazu, noch einmal über die folgende Anforderung nachzudenken:

Das SHS muss BewohnerInnen die Möglichkeit bieten, eine Tür ohne manuelle Bedienung zu entriegeln. (UC1)

Mit dieser Anforderung würde es z. B. Dieter, dem Vater von Lina, nicht gestattet sein, diese Funktion des SHS zu verwenden. An dieser Stelle sollte man also nicht nur den BewohnerInnen, sondern besser „autorisierten Personen" die automatische Entriegelung erlauben. Weiterhin kann man den Begriff Tür hinterfragen. Die Entriegelung wird sich normalerweise nur auf eine Haustür und nicht auf jede Tür des Hauses beziehen. Somit ändert sich der obige Use-Case zu:

Das SHS muss autorisierten Personen die Möglichkeit bieten, eine Haustür ohne manuelle Bedienung zu entriegeln. (UC1')

Überflüssige Anteile in einer Anforderung lassen sich häufig zwei Gruppen zuordnen: Zum einen beschreiben sie Begründungen für den eigentlich geforderten Teil und zum anderen werden Erläuterungen gegeben. Beide Anteile sollten Sie von der eigentlichen Anforderung separieren, indem Sie diese z. B. als Kommentar zu der Anforderung beschreiben.

Wir könnten an dieser Stelle noch viele weitere Beispiele zeigen, verweisen aber stattdessen auf Kapitel 9 „Das SOPHIST-*RE*gelwerk", das die Verbesserung einzelner Anforderungen im Fokus hat. Zusätzlich hilft Ihnen die Formulierung nach der MASTeR-Schablone (siehe Kapitel 19 „Schablonen für Anforderungen und User-Storys"), die Qualität einer Anforderung zu verbessern.

12.3 Angemessener Einsatz der Tätigkeiten

In dem vorigen Abschnitt haben wir Ihnen die einzelnen Aufgaben einer Analyse von Anforderungen erläutert, um diese bezüglich einiger der in Kapitel 3 „Requirements-Engineering im Überblick" eingeführten Qualitätskriterien zu verbessern. Dabei haben wir keine Rücksicht auf den dazu benötigten Aufwand genommen. Stattdessen sind wir davon ausgegangen, dass Sie eine Spezifikation erstellen wollen, die zu 100 % die hier betrachteten Qualitätskriterien erfüllt.

In der Realität eines Projekts ist es allerdings selten der Fall, dass Sie die Zeit zur Verfügung haben, alle Qualitätskriterien zu 100 % zu erfüllen. Deshalb geben wir Ihnen in diesem Abschnitt einige Tipps, wie Sie die Aufgaben gemäß Ihrer Aufgabenstellung priorisieren können, um eine für Sie optimale Anforderungsspezifikation zu erhalten.

Die Anforderungsanalyse erzeugt aus den Ursprungsanforderungen die Systemanforderungen in der benötigten Qualität.

Dazu werden wir zunächst eine Zuordnung von den Qualitätskriterien zu den einzelnen Aufgaben angeben. Damit können Sie selbst entscheiden, welche Aufgaben Sie bis zu welchem Grad durchführen möchten. Danach schlagen wir ein Vorgehen für die Analyse von Anforderungen vor, mit dem Sie zielgerichtet Ihre Ressourcen einsetzen können.

12.3.1 Die richtige Qualität erzeugen

Das Ziel bei der Analyse der Ursprungsanforderungen muss es sein, aus diesen Eingaben belastbare Systemanforderungen herzuleiten.

Oder anders formuliert: Mit jeder der zuvor vorgestellten Aufgaben werden die Anforderungen bezüglich einiger Qualitätskriterien verbessert. Wir können also eine Tabelle angeben, die die Aufgaben mit den Kriterien in Verbindung setzt. Nach der Festlegung, welche Qualitätskriterien für Sie besonders wichtig sind, können Sie damit dann auswählen, welche Aufgabe Sie fokussiert durchführen sollten und in welche Aufgabe Sie weniger Aufwand investieren können.

In der Tabelle aus Abbildung 12.8 bedeutet ein Kreuz in einer Zelle, dass die entsprechende Aufgabe das jeweilige Qualitätskriterium unterstützt. Zum besseren Verständnis wird dieser Zusammenhang im Einzelnen vorgestellt.

	Anforderungen separieren	Notwendige Anforderungen extrahieren	Anforderungen abstrahieren	Fehlende Anforderungen ergänzen	Anforderungen verfeinern	Anforderungen verbessern
Notwendig	X	X				X
Eindeutig	X				X	X
Vollständig	X					X
Vollständige Spezifikation			X	X	X	
Klare Struktur	X		X		X	
Konsistente Spezifikation	X		X		X	
Angemessenheit	X		X	X	X	

Abbildung 12.8: Zuordnung der Aufgaben zu Qualitätskriterien

Notwendige Anforderungen erhält man in erste Linie dadurch, dass man die Anforderung für den richtigen Betrachtungsgegenstand formuliert. Aber auch die überflüssigen Anteile zu eliminieren hilft Ihnen, sich auf das Notwendige in einer Anforderung zu fokussieren. Indirekt hilft dabei auch das Separieren von Anforderungen, da Sie dann die einzelnen Teile getrennt voneinander auf ihren Betrachtungsgegenstand untersuchen können.

Die **Eindeutigkeit** erreichen Sie durch die Verwendung einheitlicher Begriffe sowie durch das Separieren von Anforderungen, da hierdurch etwa die Bedingungen eindeutig einer Funktionalität zugeordnet werden müssen. Bei den Anforderungen, die nur sehr schwer eindeutig zu beschreiben sind (z. B. viele nicht-funktionale Anforderungen), hilft eine Ersetzung durch die verfeinerten Anforderungen, um sie testbar zu beschreiben.

Die **Vollständigkeit einer Anforderung** wird durch eine für einzelne Funktionen getrennte Betrachtung der Bedingungen unterstützt. Die abschließende Verbesserung einer Anforderung hilft, fehlende Bedingungen zu identifizieren.

Die **Vollständigkeit der gesamten Spezifikation** wird in erster Linie durch die Verfeinerung und Definition von fehlenden Anforderungen gegeben. Die Verfeinerung wird durch Bilden der Abstraktionen unterstützt, da hierdurch unter anderem die Basis für die Vervollständigung auf der obersten Ebene der Anforderungen gelegt wird.

Durch die Abstraktion und Verfeinerung wird auch eine **klare Strukturierung** der Anforderungen ermöglicht, da hierdurch der inhaltliche Zusammenhang der Anforderungen als Struktur der Spezifikation dienen kann (siehe auch den STABLE-Ansatz in Kapitel 21 „Strukturen und Zustände“). In zweiter Linie dient auch die Separierung von Anforderungen diesem Ziel, da die unterschiedlichen Anteile einer Ursprungsanforderung in unterschiedliche Teile der Struktur einsortiert werden können.

Die Herstellung der **Konsistenz der Anforderungen** untereinander wird unterstützt durch das Bilden des Zusammenhangs zwischen den Anforderungen im Rahmen der Abstraktion und Verfeinerung. Darüber hinaus hilft die Verwendung einheitlicher Begriffe Inkonsistenzen zu entdecken, die dann aufgelöst werden können. Aber auch die Separierung führt dazu, die Teile einer gegebenen Anforderung auf Konsistenz zu anderen Anforderungen zu prüfen.

Die **Angemessenheit der Spezifikation** wird durch das Zusammenspiel von Abstraktion und Verfeinerung unterstützt, da Sie dadurch die Möglichkeit haben, für jede Anforderung zu entscheiden, ob Sie an dieser Stelle noch mehr Aussagen treffen müssen oder nicht. Da der Aufwand für die Anforderungsanalyse gerade hierdurch primär bestimmt wird, wollen wir diese Entscheidung in dem nachfolgenden Abschnitt gesondert betrachten.

12.3.2 Was wirklich benötigt wird

Nachdem wir Ihnen die einzelnen Aufgaben anhand von Beispielen erläutert haben und deren Bezug zur Qualitätsverbesserung Ihrer Systemanforderungen beschrieben haben, geben wir Ihnen nun Vorgehen für die Anwendung in der Praxis an die Hand. Dazu möchten wir Szenario 1 „Kundenanfrage bearbeiten“ und Szenario 3 „Subunternehmen beauftragen“ aus Kapitel 4 „RE ist nicht gleich RE“ verwenden.

Im zuerst genannten Szenario antwortet die Firma Schlauhaus mit einem Angebot in Form eines Pflichtenheftes auf die Anfrage von Familie Meyer. In diesem Fall sind die Ursprungsanforderungen in einem Lastenheft enthalten. Im Szenario 3 „Subunternehmen beauftragen“ dagegen erstellt Schlauhaus ein eigenes Lastenheft für eine Ausschreibung. Dementsprechend handelt es sich bei den Ursprungsanforderungen um Anforderungen von den Stakeholdern aus der Firma Schlauhaus.

Für die Analyse der Anforderungen gehen wir in beiden Fällen davon aus, dass eine Spezifikation erstellt und eine Nachvollziehbarkeit zu den entsprechenden Eingaben (also Stakeholder- oder Lastenheftanforderungen) hergestellt werden soll, wie es in einem

regulierten Umfeld oder von vielen Entwicklungsprozessen verlangt wird. Daraus lassen sich die folgenden zwei Qualitätsstufen für eine Spezifikation ableiten:

- **Minimale** Spezifikation: Hier werden genau die Anforderungen definiert, deren Begründungen sich von den Ursprungsanforderungen ableiten lassen. Die hier beschriebenen Anforderungen beschreiben nur das Nötigste und lassen der weiteren Entwicklung, zum Beispiel dem Auftragnehmer, sehr viele Freiheiten.
- **Vollständige** Spezifikation: Aufbauend auf einer Spezifikation mit minimaler Qualität (minimale Spezifikation) werden weitere Anforderungen definiert, die den Gesamtumfang des zu entwickelnden Systems beschreiben (vollständige Spezifikation).

Sie können auch die Erstellung der minimalen Spezifikation als Pflichtprogramm ansehen, da sie das Mindeste darstellt, das in einer Anforderungsanalyse entstehen sollte. Auf dem Weg zur vollständigen Spezifikation haben Sie dann die Wahl, wie viel Aufwand Sie investieren bzw. welchen Grad von Vollständigkeit Sie erreichen möchten. Demnach wird sich Ihre Spezifikation in Ihrem Projekt irgendwo zwischen diesen beiden Extrema bewegen.

Erzeugung einer minimalen Spezifikation

Wie eingangs erwähnt, möchten wir zunächst die Anforderungen finden, die ihre Basis in den Anforderungen haben, die Ihnen als Requirements-Engineer als Eingabe, z. B. von ihren Stakeholdern oder in einem Lastenheft, vorliegen. Die einfachste Art zur Erzeugung einer minimalen Spezifikation wäre es, die Ursprungsanforderungen einfach zu übernehmen. Um die weitere Bearbeitung der Anforderungen zu unterstützen, wollen wir allerdings die Qualität der Anforderungen in den Bereichen *Notwendigkeit* und *Eindeutigkeit* steigern. Des Weiteren soll die Spezifikation *klar strukturiert* sein, auch um die Erweiterung der minimalen Spezifikation hin zu einer vollständigen Spezifikation zu ermöglichen.

Die hierfür benötigten Tätigkeiten bei der Analyse der Anforderungen sind in Abbildung 12.9 dargestellt. Bitte beachten Sie, dass wir hier weder die Schritte zur Dokumentation noch zur Prüfung der gefundenen Anforderungen adressiert haben. Wir beziehen uns nur auf die zuvor definierten Aufgaben.

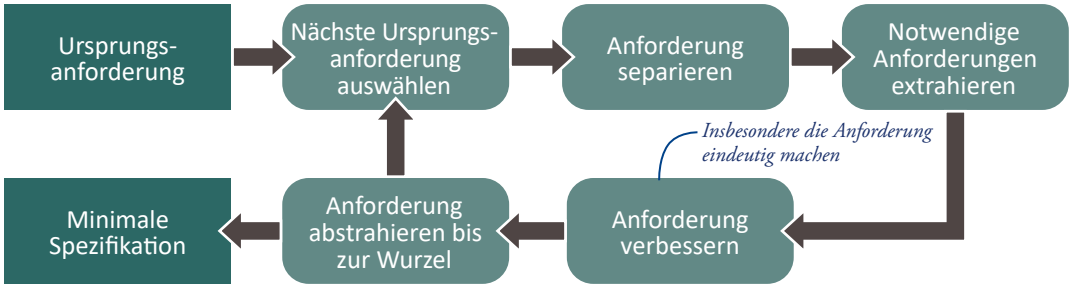


Abbildung 12.9: Erzeugung einer minimalen Spezifikation

Bitte beachten Sie auch, dass Sie auf jeden Fall die Aufgabe *Notwendige Anforderungen extrahieren* durchführen sollten. Bei den nachfolgenden Aufgaben können Sie bei Bedarf Abstriche machen.

Wenn Sie die dargestellten Schritte für jede der Ursprungsanforderungen durchgeführt haben, so werden Sie schon ein gutes Gefühl haben, was Ihre Stakeholder von dem zu entwickelnden System erwarten.

Erweiterung zu einer vollständigen Spezifikation

In dieser zweiten Phase wollen wir nun die zuvor erzeugte minimale Spezifikation zu einer vollständigen Spezifikation erweitern. Die Ursprungsanforderungen müssen nun nicht mehr betrachtet werden, da sie in der Phase zuvor vollständig in neue Anforderungen überführt wurden.

Zunächst sollten Sie sowohl die Use-Cases als auch die Kategorien nicht-funktionaler Anforderungen auf Vollständigkeit überprüfen. Danach können Sie mit einem iterativen Prozess starten: Wählen Sie die risikoreichste Anforderung aus und führen Sie die angegebenen Schritte durch. Danach haben Sie weitere Anforderungen gefunden, die nun für die weitere Bearbeitung zur Verfügung stehen.

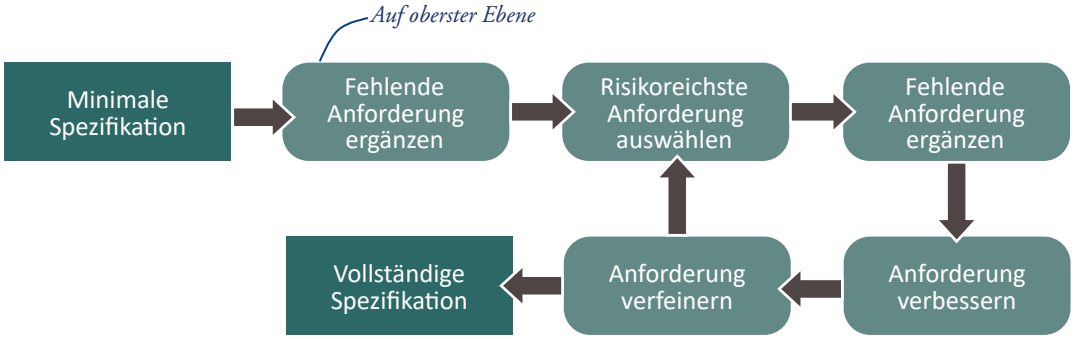


Abbildung 12.10: Erzeugung einer vollständigen Spezifikation

Mit dem hier angegebenen Vorgehen sind Sie in der Lage, Ihre Ressourcen zur Anforderungsanalyse zielgerichtet zu investieren und immer wieder bewusst zu entscheiden, an welcher Stelle Ihrer Anforderungsspezifikation Sie Aufwand zur Qualitätssteigerung investieren möchten.

Chris Rupp, Carsten Pflug

17

Storytelling, User-Storys und Co. – verschiedene Arten, Anforderungen zu vermitteln

In diesem Kapitel beschreiben wir einige Möglichkeiten der Vermittlung von Anforderungen. Dabei ist den Techniken gemein, dass sie ohne eine Anforderungsdokumentation im klassischen Sinne auskommen. Viele davon (z. B. User-Storys oder Storytelling) werden vor allem in der agilen Welt häufig eingesetzt.

Allerdings schließen sich Anforderungsdokumentation mit Modellen oder natürlichsprachliche Anforderungsdokumentationen und die hier vorgestellten Möglichkeiten nicht gegenseitig aus. Vielmehr können Sie diese durchaus kombinieren, um die Stärken beider Varianten zu vereinen.

Möchten Sie wissen, wann welche Techniken am sinnvollsten einzusetzen sind, lesen Sie Kapitel 16 „Wegweiser: Anforderungen dokumentieren und vermitteln“. Hier finden Sie auch die Einflussfaktoren, anhand derer Sie eine Eignung der einzelnen Ansätze für die drei Szenarien aus Kapitel 4 „RE ist nicht gleich RE“ beurteilen können.

Wir möchten Sie darauf hinweisen, dass nicht alle Techniken, die Sie zum Vermitteln verwenden können, in dem vorliegenden Kapitel zu finden sind. Eine Erläuterung, in welchen Kapiteln Vermittlungstechniken beschrieben sind, finden Sie in Kapitel 16 „Wegweiser: Anforderungen dokumentieren und vermitteln “.

17.1 Storytelling – Grimms Märchen der Anforderungsvermittlung

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.

Definition Storytelling [Mattschek]

Storytelling beschreibt eine Kommunikationsart zur Vermittlung von Informationen, Wissen, Werten, Meinungen etc. Dies kann über Sprache, Text, Bild oder Videos erfolgen. Dabei werden nicht emotionale Inhalte in Geschichten verpackt, um über die Geschichte Emotionen und Interesse bei Zuhörern, Lesern oder Betrachtern zu wecken.

Storytelling eignet sich im Requirements-Engineering vor allem für das Ermitteln und das Vermitteln von Visionen, Zielen, Epics, Geschäftsprozessen, Personas, Anforderungen und User-Storys. Beim Ermitteln von Wissen wird meist der Stakeholder die Geschichten erzählen. Beim Vermitteln von Wissen können sowohl der Product-Owner, der Requirements-Engineer, oder aber auch ein Stakeholder Wissen in Form einer Story an das Entwicklungsteam oder den/die AuftragnehmerIn vermitteln. Der Product-

Owner und/oder Requirements-Engineer sollte lernen, wie man Wissen effektiv mittels Storys vermittelt. Wir gehen bei den Stakeholdern nicht davon aus, dass sie perfekte Storyteller sind. Wir leiten sie durch Rückfragen und Hinweise dazu an, die wichtigsten Regeln des Storytellings einzuhalten. Wenn z. B. ein Stakeholder nicht von sich spricht, sondern immer wieder in Formulierungen verfällt wie „man braucht doch …, man macht doch …“, dann hinterfragen Sie einfach: „Brauchen Sie diese Funktion?“, „Wie gehen Sie denn mit der Situation um?“.

Als Beispiel dient uns hier eine Geschichte von Lina, in der sie uns ihre Wünsche zum automatischen Öffnen und Schließen der Zugangstür erzählt:

Lina Meyer:

„Sonntags liebt es meine Familie nachmittags bei einer Tasse Tee und einem leckeren Stück Kuchen – gerne auch mal mit Freunden zusammen – im Garten zu entspannen. Dazu fahre ich nach dem Mittagessen mit dem Auto zu unserer lokalen Bäckerei. Von dort komme ich mit mehreren Kuchenpäckchen auf dem Arm zurück. Voll beladen stehe ich dann vor der Eingangstür. Ich versuche dann den Hausschlüssel aus der Tasche zu kramen, ohne den Kuchen auf dem Boden ablegen zu müssen, was meist nicht funktioniert. Klingeln bringt auch nix, denn die Familie befindet sich ja im Garten. Das neue Türöffnungssystem sollte mich erkennen, mir in Zukunft die Tür automatisch öffnen und sie hinter mir wieder schließen.“

17.1.1 Arten des Storytellings

Es gibt unterschiedliche Arten von Storys [Löhr], deren Einsatz im Requirement-Engineering für die Vermittlung von Wissen Sinn ergibt.

Hintergrundstorys erzählen etwas über die Umgebung und die Nutzung des Systems – sie übermitteln wichtige Hintergrundinformationen. Sie explizieren damit oftmals implizites Wissen, welches den Zuhörenden dabei hilft die Anforderungen besser zu verstehen. In unserem Beispiel würde z. B. ein Video über das Haus der Familie Meyer und die Umgebung des Hauses Wissen zur Geografie, der Anmutung und der Historie des Anwesens vermitteln und damit ein erstes Bild der Situation in den Köpfen aller Beteiligten entstehen lassen. Das Kapitel 27 „Videos im RE“ zeigt zum Beispiel, wie ein Umgebungsvideo die Einsatzumgebung des Smart-Home-Systems (SHS) vermitteln kann.

Personality-Storys stellen die Personas oder Stakeholder des Systems vor und machen deren Eigenschaften und Persönlichkeit erlebbar. Neben den Fakten zur Person (wie Alter, Beruf, Ausbildung) zeigt eine Personality-Story konkrete Meinungen, Vorlieben, Einstellungen zu relevanten Themen (z. B. Umgang und Einstellung zu IT-Systemen). Im Vergleich zu einer schriftlichen Personabeschreibung oder den Einträgen

in einer Stakeholderliste ist z. B. ein Video zur Vorstellung einer konkreten Person (eines Stakeholders) oder einer fiktiven Persona, die eine Geschichte über sich erzählt, viel informationsreicher und einprägsamer. Neben dem gesprochenen Text wird sehr viel Information über das Temperament, den Charakter, die Art zu interagieren und zu sprechen der gefilmten Person vermittelt. In unserem Beispiel würden Personality-Videos der BewohnerInnen mit den jeweiligen Einstellungen und Charakteren einen guten Eindruck der zukünftig Nutzenden vermitteln.

Achten Sie darauf, dass Personality-Videos als Minimalinhalt die Informationen enthält, die auch in der Stakeholdercheckliste vermerkt würden oder in der Personabeschreibung enthalten wären, falls das Video diese Artefakte ersetzen soll.

Überzeugungsstorys transportieren ganz am Anfang des Requirements-Engineerings die Ausgangslage und erklären motivierend, warum das System gebraucht wird. Im Beispiel unseres SHS könnte z. B. die Architektin in einem ambitionierten Plädoyer die Vision des neuen smarten Hauses der Familie Meyer vermitteln. Ein Beispiel für eine Überzeugungsstory für ein Smart Home finden Sie am Anfang des Kapitels in Abschnitt 17.1 „Storytelling – Grimms Märchen der Anforderungsvermittlung“. Achten Sie bei Überzeugungsstorys darauf, dass die Ziele/Visionen des Systems kommuniziert werden, ohne die Realisierung vorzugeben (sofern diese noch nicht feststeht). Ein Beispiel für eine Überzeugungsstory für das Requirements-Engineering sehen Sie in folgendem Video:

Erklärungsstorys verdeutlichen einzelne Abläufe und Verhaltensweisen des Systems. Wenn komplexe Sachverhalte mit Bildern und Geschichten untermalt werden, können Sie das Ganze besser nachvollziehen und sich merken. Auch Ihre Aufmerksamkeit wird durch Visualisierung (sei es auch nur im Kopf) stärker angesprochen. Wie wäre es für unser konkretes Beispiel mit mehreren Videos oder einer Erzählung, die verdeutlicht, wie sich die Familie Meyer später das automatisierte Öffnen ihrer Haustür vorstellt? Eine Erklärungsstory muss alle relevanten Vor- und Nachbedingungen des Ablaufs enthalten, ebenso wie die Prozessschritte, sofern diese schon bekannt sind. Eine Erklärungsstory ist somit ein natürlichsprachliches Szenario, das sich hervorragend eignet, um die gewünschte Interaktion des Erzählers mit dem System zu verstehen. Daraus lassen sich im nächsten Schritt Anforderungen, z. B. in Form von User-Storys, ableiten. Szenarien helfen auch dabei Abläufe systematisch zu durchdenken – insbesondere wenn der Stakeholder lieber in konkreten Beispielen statt in abstrakten Anforderungen denkt und spricht. Weitere Infos zum Thema Szenarien finden Sie unter www.sophist.de/re7/kapitel17.

17.1.2 Was macht gutes Storytelling aus?

Damit eine Story für die Vermittlung geeignet ist, muss sie gut sein. Aber was heißt das konkret für den Einsatz im Requirements-Engineering? Die Botschaft der Geschichte muss eindeutig sein (denn uns geht es ja um das vermittelte Wissen). Zudem müssen sowohl die Geschichte als auch die Person, die die Geschichte erzählt, authentisch wirken (damit der Vermittlungsprozess optimal funktioniert). Das lässt sich durch die folgenden Punkte erreichen [Fordon17]:

- Die Story muss nicht unbedingt wahr sein, aber reale Storys lassen sich authentischer erzählen als erfundene. Eine erfundene Geschichte sollte die erzählende Person gut vorbereiten, sodass es dem Publikum gar nicht auffällt, dass sie erfunden ist.
- Die Person, die erzählt, verstellt sich nicht, sondern verhält sich genauso, wie sie es auch privat tun würde. Falls Sie als Product-Owner oder Requirements-Engineer eine Geschichte wiedergeben, die Ihnen von Ihrem Stakeholder erzählt wurde, so stellen Sie einfach die Info voran, wer Ihnen aus welchem Anlass diese Geschichte erzählt hat, und geben diese möglichst genau wieder. Hier eignen sich z. B. Videoaufnahmen von Geschichten.
- Die Geschichte ist sprachlich und inhaltlich an die Zielgruppe angepasst.
- Drei bis fünf Minuten sind eine gute Erzähldauer [Oswald14].
- Bei den Zuhörenden müssen Emotionen geweckt werden durch [Fordon17]:
  - emphatische Charaktere,
  - eine schlüssige Hintergrundgeschichte,
  - adaptierbare Werte und Moralvorstellungen eines Charakters,
  - Ziele, die ein Charakter oder die Geschichte verfolgt,
  - Gefahren oder Konflikte, die überwunden werden (auch wenn das im RE meist nur kleine sind).

17.1.3 Die irrelevanten Teile einer Story

Eine Story enthält immer Informationen, die nicht direkt etwas mit dem System zu tun haben und auch nicht für dieses relevant sind. Allerdings werden diese Informationen benötigt, um den Inhalt der Story besser zu verstehen oder eine authentische Geschichte zu haben, die sich gut merken lässt. In der oben beschriebenen Story für das Smart Home von Lina zum automatischen Öffnen der Eingangstür ist beispielsweise die Information, dass Lina zur Bäckerei fährt und die Arme und Hände mit Kuchen beladen hat, wenn sie vor der Tür steht, irrelevant. Ausreichend wäre die Information, dass eine registrierte Person die Arme nicht zum Öffnen und Schließen der Tür verwenden kann. Die Story lässt sich allerdings besser merken, da sie Bilder im Kopf der Zuhörenden erzeugt.

Es kommt zu einem sogenannten „medialen Kollateralinhalt“, also einem Inhalt, der entsteht und nicht gebraucht wird, aber in Kauf genommen wird, um den relevanten Inhalt erfolgreich zu vermitteln. Denken Sie darüber nach, wie groß dieser „irrelevante Teil“ sein muss, um den Kern des relevanten Inhalts zu erzählen, und gestalten Sie ihn möglichst minimal.

17.1.4 Gute Geschichten für eine gute Vermittlung

Storytelling ist eine sehr effektive Möglichkeit, um Wissen zu vermitteln. Da die Entwicklung einer guten Story aufwendig ist, sollten Sie sich der Vor- und Nachteile des Storytellings bewusst sein.

- **Storys erzeugen mehr Aufmerksamkeit als sachliche Vorträge**
- **Storys entspannen die Situation, da wir uns unbewusst daran erinnern, wie uns als Kind vorgelesen wurde. Diese positive Erinnerung beruhigt uns.**
- **Durch Storys können Anforderungen besser vermittelt werden, denn sie helfen komplexe Sachverhalte auf einfachere Ebenen zu transferieren. Storys erzeugen Bilder in unseren Köpfen. Durch Bilder können wir Sachverhalte besser verstehen.**
- **Storys wecken Emotionen und sorgen dafür, dass Inhalte länger im Kopf bleiben und weitervermittelt werden.**
- **Die Zuhörenden benötigen keine speziellen Kenntnisse, um einer Story folgen zu können.**

- **Das Entwickeln geeigneter Storys ist aufwendig.**
- **Die erzählende Person muss authentisch wirken und gut kommunizieren können.**
- **Storys sind schwer wartbar und somit zur Konservierung von Wissen weniger geeignet.**
- **Storys beinhalten immer Informationen, die inkonsistent sind (zu anderen Storys) oder irrelevant sind (und nur der Erzählung dienen).**

Als Requirements-Engineer sollten Sie wissen, wie man eine gute Story aufbaut und wie man diese wirkungsvoll erzählt. Stakeholder haben dieses Wissen meist nicht. Hier sollten Sie den Stakeholder durch geschickte Fragestellung oder Hinweise in die richtige Richtung lenken.

Falls Sie Storys nicht nur erzählen, sondern anderweitig präsentieren wollen, müssen Sie sich Gedanken über die Dokumentation bzw. Konservierung von Storys machen. Hier eignen sich Videos deutlich besser als die Verschriftlichung der Geschichten, da in Videos Mimik und Gestik der Erzählenden zu sehen sind. Detaillierte Informationen, wie man Storys mittels Videos dokumentiert und was dabei zu beachten ist, erhalten Sie in Kapitel 27 „Videos im RE“.

Für die Wartung eignen sich Anforderungen oder Abläufe ohne Rahmenstory allerdings weit besser. Mehr dazu finden Sie in Kapitel 16 „Wegweiser: Anforderungen dokumentieren

und vermitteln“. Somit sollten Sie sich überlegen, ob es wichtiger ist, die Informationen erfolgreich zu vermitteln, oder diese auch zu verwalten.

17.2 User-Storys und Story Mapping

User-Storys beschreiben gewünschte Funktionalitäten bzw. Eigenschaften (siehe Kapitel 19 „Schablonen für Anforderungen und User-Storys“) eines Systems aus der Sicht derjenigen Person, welche die Funktionalität bzw. Eigenschaft benötigt. Dabei muss diese Person nicht unbedingt diejenige sein, welche das System letztendlich benutzen wird. Nehmen wir als Beispiel folgende User-Story:

Als GesetzgeberIn möchte ich, dass die Daten über Wohnungszutritte nach x Wochen gelöscht werden, um die Daten von Personen zu schützen.

Die Funktionalität aus diesem Beispiel wird nicht von den nutzenden Personen des Systems gefordert, sondern von jemand anderem, hier der dem Gesetzgeber oder der Gesetzgeberin.

In der agilen Welt ist die Vermittlung von Anforderungen mittels User-Storys weitverbreitet.

Definition User-Story nach Cohn [Cohn10]

Eine User-Story beschreibt eine Funktionalität, die für den Kunden oder Benutzer eines Produkts oder Systems von Wert ist. Sie besteht aus der schriftlichen Beschreibung der Funktionalität, Gesprächen über die Funktionalität und Akzeptanzkriterien, die Details vermitteln und festlegen, wann eine Story vollständig umgesetzt ist.

Angelehnt an diese Definition wurde eine Satzschablone für die Formulierung von User-Storys entwickelt. Diese ist in Abschnitt 19.6 „Schnell und einfach zur User-Story“ beschrieben.

Entsprechend könnte eine User-Story lauten:

Als BewohnerIn möchte ich eine Kameraüberwachung rund ums Haus, um während meiner Abwesenheit nachprüfen zu können, ob sich Personen auf meinem Grundstück befinden.

Sie werden aber sicherlich an der einen oder anderen Stelle auf das Problem stoßen, dass Sie Ihre User-Story nicht mit dieser Schablone formulieren können oder das formulierte Ergebnis komisch klingt. In solchen Fällen kann es sein, dass für diese User-Story nicht alle Punkte aus der obigen Definition zutreffen. Nehmen Sie einfach ein paar Anpassungen vor. Das könnte zum Beispiel sein, das Template für User-Storys abzuändern oder gar ganz ohne Satzschablone zu formulieren.

17.2.1 Verschiedene Detaillierungsebenen von User-Story – von Epics bis zu detaillierten User-Storys

Die in der User-Story beschriebene Funktionalität bzw. Eigenschaft kann auf unterschiedlichen Verfeinerungsgraden liegen (siehe Kapitel 3 „Requirements-Engineering im Überblick“). So könnte eine User-Story zum Beispiel „Schlüssellosen Zugang zum Haus“ als Thema haben, oder aber eher detailliert die Funktionalität „Erkennen einer autorisierten Person“ spezifizieren. Für sehr grobe User-Storys wird manchmal der Begriff Epic verwendet. Allerdings gibt es keine konkreten Richtlinien, wann eine User-Story ein Epic ist und wann nicht. Deshalb sprechen wir im Weiteren nur von User-Storys.

Grobe User-Storys haben durchaus ihre Daseinsberechtigung, zum Beispiel um detaillierte Storys einordnen zu können oder um einen Überblick zu bekommen. Allerdings gibt es Gründe eine grobe User-Story in detailliertere zu zerlegen. Man spricht in diesem Fall vom Schneiden der User-Storys. Warum grobe User-Storys in feine zerlegt werden und wie weit man dabei geht, lesen Sie in Kapitel 12 „Anforderungen analysieren“. Dort ist das Konzept für Anforderungen und User-Storys beschrieben. Zusätzlich finden Sie im Internet unter www.sophist.de/re7/kapitel17 weitere Tipps und Tricks zum Schneiden von User-Storys.

Als BewohnerIn möchte ich einen schlüssellosen Gebäudezugang, um mit vollgepackten Händen das Gebäude betreten zu können.

Diese User-Story ist sehr wahrscheinlich noch zu grob, um sie in einer Iteration umsetzen zu können, und für das Entwicklungsteam nur schwer abschätzbar. Dementsprechend sind nicht alle Kriterien des INVEST-Prinzips aus Kapitel 3 „Requirements-Engineering im Überblick“ erfüllt. Auch können detaillierte User-Storys hier und da gegen Teile dieses Prinzips verstoßen. Zum Beispiel haben Sie möglicherweise User-Storys, die nicht komplett unabhängig von anderen User-Storys sind. Sie sollten auf jeden Fall darauf achten, dass eine User-Story testbar und schätzbar ist.

17.2.2 Vermitteln mit User-Storys

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, und dann ein gemeinsames Zustimmen über die Inhalte der User-Story,

Abbildung 17.1: 3-C-Modell

Das Konzept wird üblicherweise als die 3 Cs – Card, Conversation und Cofirmation – bezeichnet. Es beginnt mit der Karte, auf der die User-Story mit den Akzeptanzkriterien formuliert wird. Dann folgt das gemeinsame Gespräch über die User-Story. An dessen Ende sagen dann alle Parteien, dass sie die User-Story verstanden haben und diese so umgesetzt wird.

17.2.3 Formulieren einer User-Story

Prinzipiell sind Sie frei, was die Formulierung einer User-Story betrifft. Als Hilfsmittel für die Formulierung können Sie ein Template für User-Storys verwenden, welches im Abschnitt 19.6 „Schnell und einfach zur User-Story“ beschrieben ist. Wie aber bereits geschildert, sollten Sie den Einsatz einer Schablone nicht als in Stein gemeißeltes Gesetz ansehen. Mit der Satzschablone formuliert, könnte die User-Story folgendermaßen lauten:

Als BewohnerIn möchte ich, dass das Haus automatisch verriegelt wird, wenn sich niemand im Haus befindet, um sicherzugehen, dass das leere Haus immer verriegelt ist.

Es hat sich in der Praxis bewährt, immer wieder über die Formulierung seiner User-Storys nachzudenken und entsprechend dem Feedback, das man beim Einsatz von User-Storys erhält, die Art der Formulierung anzupassen. Das Gleiche gilt auch für die Inhalte, die in einer User-Story beschrieben sind. Es kann sein, dass ein einzelner Satz für Ihre User-Story vollkommen ausreichend ist, sodass jeder weiß, was zu tun ist. Es kommt aber auch vor, dass in einer User-Story viel mehr beschrieben werden muss. Letzteres kann vor allem dann der Fall sein, wenn Sie als die Person mit dem Wissen nicht oft für Nachfragen erreichbar sind oder der Inhalt der User-Story etwas Neues, bisher Unbekanntes für die WissensempfängerInnen beschreibt.

Zu einer User-Story gehören Akzeptanzkriterien. Akzeptanzkriterien legen fest, wann eine User-Story vollständig umgesetzt ist. Sie beschreiben mit den Akzeptanzkriterien, was gegeben sein muss, damit Sie die User-Story als umgesetzt betrachten können. Auch für Akzeptanzkriterien gibt es keine feste Vorgabe, wie diese formuliert werden müssen, aber eine Schablone, die Sie nutzen können. Diese Schablone finden Sie in Abschnitt 19.6.2 „Aufbau und Inhalt von Akzeptanzkriterien für User-Storys“. Sie sollten bei der Formulierung darauf achten, dass aus den Akzeptanzkriterien klar hervorgeht, wie und was Sie prüfen, um eine Aussage treffen zu können, ob die User-Story zu Ihrer Zufriedenheit umgesetzt wurde. Ein Akzeptanzkriterium für unsere User-Story könnte folgendermaßen lauten:

Angenommen: Im Haus befindet sich eine Person

Wenn: Die Person verlässt das Haus

Dann: Die Tür ist verriegelt

17.2.4 Das Gespräch zu einer User-Story

Eine wichtige Tätigkeit beim Einsatz von User-Storys ist das gemeinsame Gespräch zur Story. In diesem Gespräch werden die detaillierten Inhalte der User-Story vermittelt. Man sagt häufig, User-Storys sind ein Kommunikationsversprechen. Dementsprechend gehört das Sprechen über eine Story zu User-Storys dazu. Für dieses Gespräch ist es wichtig, dass Sie sich entsprechend vorbereiten.

Für die Vorbereitung sammeln Sie alles Material zusammen, welches Ihnen im Gespräch hilfreich sein kann. Das können Bilder, Diagramme, Prototypen und so weiter sein. Denken Sie daran, möglichst viele Sinne anzusprechen, also nicht nur den Hörsinn. Des Weiteren sollten Sie prüfen, ob Sie ausreichend Wissen zu einer User-Story haben, um alle möglichen Fragen beantworten zu können. Hierzu könnte das *RE*gelwerk aus Kapitel 9 „Das SOPHIST-*RE*gelwerk“ hilfreich sein. Anhand der Regeln können Sie für sich nachprüfen, ob Ihnen noch Informationen zu der User-Story fehlen, oder ob sie mögliche Fragen konkret genug beantworten können.

Für das Gespräch sollten Sie sich überlegen, ob die User-Story, über die gesprochen wird, eine vernünftige Größe hat. Damit ist der Umfang des Inhalts gemeint. Die User-Story „Die nutzende Person möchte schlüssellosen Zugang zum Haus haben“ bietet sehr viel Gesprächsstoff und ist daher für ein Gespräch, bei dem alle Beteiligten das Problem vollständig durchdringen sollen, eher ungeeignet. Da tun Sie sich mit einer User-Story der Größenordnung „Die Tür soll beim Entriegeln ein akustisches Signal abgeben“ viel leichter.

Für das Gespräch selbst ist es wichtig, gerade bei sehr detaillierten User-Storys, den Personen, die das Wissen empfangen, die Einordnung der User-Story in deren Kontext zu ermöglichen. Dazu haben Sie verschiedene Mittel. Sie könnten zum Beispiel mittels Storytelling oder mit einer Story Map die Einbettung der User-Story in deren Kontext darstellen. Achten Sie auch auf ausreichende und sinnvolle Visualisierungen während des Gesprächs und halten Sie dafür die entsprechenden Mittel wie Flipcharts, Whiteboards etc. parat.

Wie Sie sehen, ist beim Einsatz von User-Storys das gemeinsame Gespräch ein zentraler Punkt. Das kann allerdings zu einem Problem führen, da das Wissen, welches sich in einem Gespräch verbreitet, schnell auch wieder vergessen wird (vergleichen Sie hierzu die Vergessenskurve in Kapitel 1 „In medias RE“). Dementsprechend ist diese Technik nicht so gut geeignet, um Wissen über einen längeren Zeitraum zu bewahren. Üblicherweise werden nur die User-Storys gemeinsam besprochen, die relativ zeitnah realisiert werden sollen. User-Storys, die erst in ferner Zukunft realisiert werden, werden nicht im Detail besprochen, sondern höchstens mal kurz angerissen, um einen Überblick zu geben, was alles noch kommen wird. Dem Problem des Vergessens können Sie entgegenwirken, indem Sie einerseits mehr Informationen in der User-Story schriftlich festhalten oder andererseits eine zusätzliche Dokumentation parallel zu den User-Storys anlegen (siehe Kapitel 16 „Wegweiser: Anforderungen dokumentieren und vermitteln“). Gerade die erstgenannte Variante ist interessant, wenn Sie als ÜbermittlerIn der Anforderungen für Nachfragen bei der Umsetzung eher sporadisch zur Verfügung stehen. Die

zweite Variante – also eine zusätzliche Dokumentation – ist dann ratsam, wenn das vermittelte Wissen auch lange nach der Umsetzung zur Verfügung stehen muss.

17.2.5 Story Mapping – das Gesamtbild betrachten

Beim Arbeiten mit User-Storys kann man schnell den Überblick über das Gesamtbild verlieren. Mitunter können Ihre User-Storys nur einen ganz kleinen Teil des zu entwickelnden Systems betreffen und dadurch ist es sehr schwer, anhand der User-Story selbst den Zusammenhang mit anderen User-Storys zu erkennen.

Um den Überblick zu behalten, kann die Technik des Story Mappings [Patton14] verwendet werden. In einer Story Map werden die User-Storys nach speziellen Aspekten angeordnet. Zum Beispiel können die User-Storys entsprechend ihrer zugehörigen Epics sortiert werden. Oder die User-Storys werden nach einem darüberliegenden Prozess sortiert. Eine andere Variante ist, die User-Storys entsprechend einer Releaseplanung zu sortieren. Je nach Information, die man aus einer Story Map gewinnen möchte, wird die Anordnung der User-Storys in einer Story Map anders sein.

Wir wollen Ihnen hier ein Beispiel einer Story Map vorstellen, mit deren Hilfe die Einordnung einer User-Story in ihren Kontext gezeigt werden kann. Dafür werden anhand von Szenarien aus Benutzungssicht die essenziellen Grundfunktionalitäten des Systems festgelegt. Diese können in Form von Epics beschrieben sein. Diese Epics werden horizontal ganz oben in einer Story Map angeordnet und bilden das sogenannte Rückgrat der Story Map (engl. backbone). Unterhalb des Rückgrats werden die User-Storys angeordnet, die wichtig für eine Minimalversion des Systems sind. Diese sollten Sie, falls möglich, entsprechend ihrer Reihenfolge im Geschäftsprozess von links nach rechts anordnen. Unterhalb der User-Storys, die die Minimalversion darstellen, werden nun alle weiteren User-Storys entsprechend ihrer Zugehörigkeit zu der Minimalversion angeordnet. Dabei ist es üblich, User-Storys mit hoher Priorität weiter oben und Storys mit niedriger Priorität weiter unten anzuordnen.

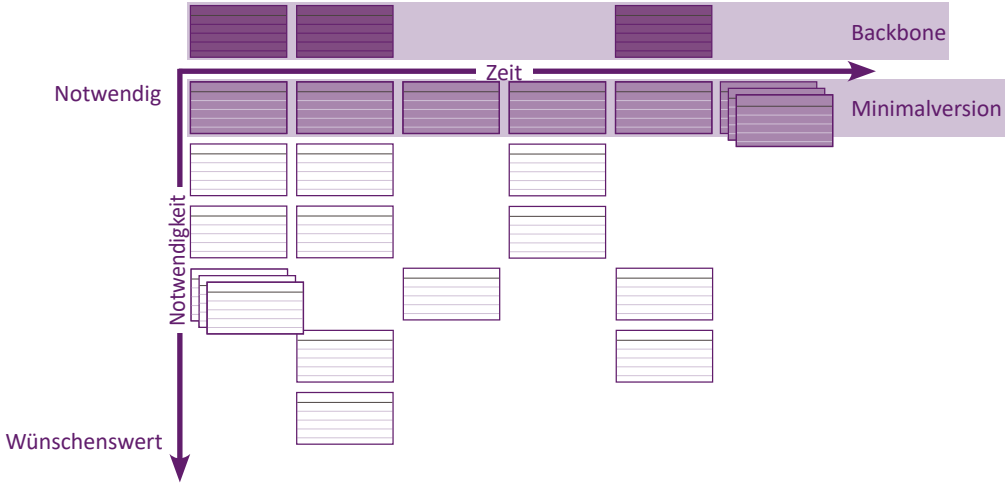


Abbildung 17.2: Grafik einer Story Map

Solch eine Story Map können Sie nutzen, um in einer Diskussion über eine User-Story den Beteiligten den Kontext dieser einzelnen User-Story zu vermitteln. Story Maps sind auch gut geeignet, um fehlende Anforderungen (hier in Form von User-Storys) aufzudecken bzw. um Anforderungen initial zu finden, und können daher auch für das Ermitteln von Anforderungen eingesetzt werden. Wie eine Story Map erstellt wird, können Sie auch in folgendem Video sehen:

17.2.6 Gute User-Storys für eine gute Vermittlung

User-Storys sind ein sehr effektives Mittel, um Wissen zu vermitteln. Da die Entwicklung aufwendig ist, sollten Sie sich der Vor- und Nachteile von User-Storys bewusst sein.

- **User-Storys entwickeln bei allen Beteiligten ein gemeinsames Verständnis zu den Anforderungen.**
- **Durch die nutzungsorientierte Beschreibung von User-Storys wird man sich der Notwendigkeit der Anforderungen bewusst.**
- **Durch das Schneiden der User-Storys nach dem INVEST-Prinzip erhalten diese eine für die Kommunikation angemessene Größe.**

- **User-Storys eignen sich nicht für die Aufbewahrung detaillierter Anforderungen.**

17.3 Prototypen – everybodys darling

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 stellen Ihnen hier die Arten vor, die wir für den Bereich der Wissensvermittlung für geeignet halten. Dabei fokussieren wir uns auf die Oberflächen-Prototypen.

17.3.1 Wireframe – das Drahtmodell für den Bildschirm

Ein Wireframe war in der Schneiderbranche eine sehr einfach Schneiderpuppe, die die Außenlinien des Körpers mittels Draht abbildete. In der Informatik spricht man von einem Wireframe, wenn es sich um einen sehr frühen Entwurf einer Benutzeroberfläche handelt, bei dem weder Gestaltung noch Details bezüglich der Funktion eine Rolle spielen. Der Fokus liegt auf der Anordnung von Elementen und der Benutzerführung. Alle

wichtigen Elemente des Endprodukts, zum Beispiel Eingabefelder oder Schaltflächen, sollten enthalten sein. Das Wireframe legt damit die Basis für die weitere Entwicklung und eignet sich für die Vermittlung der wichtigsten Elemente, die auf einer Oberfläche dargestellt werden. Wireframes werden nach ihrer Nutzung entweder Teil der Spezifikation oder sie werden verworfen.

Die folgenden beiden Arten von Wireframes werden eigenständig oder in Kombination verwendet:

- Statische Wireframes visualisieren jeweils eine einzelne Seite der Software-Benutzungsoberfläche schematisch. Sie zeigen die grundlegenden Elemente und den konzeptuellen Aufbau einer Seite.
- Dynamische Wireframes umfassen mehrere interaktiv verknüpfte Seiten, die eine Navigation von einer zur anderen Seite ermöglichen. Sie bilden somit eine einfache Form eines funktionalen Prototyps.

Da es bei einem Wireframe um die Struktur und Logik und nicht um das Design geht, genügen für die Darstellung rudimentäre grafische Elemente, was die Aufwände bei der Erstellung reduziert. Auch auf Farben wird in dieser Phase grundsätzlich verzichtet, aber durch die Verwendung unterschiedlicher Graustufen können Elemente bewusst hervorgehoben werden. Dadurch lässt sich eine gewünschte Wahrnehmungsreihenfolge aufbauen. Die einfachste Form eines Wireframes ist ein Papierprototyp oder ein Low-fidelity-prototyp, der mit einem Tool erstellt wird. Wireframes werden nach ihrer Nutzung für die Ermittlung oder Vermittlung nicht zum Endprodukt weiterentwickelt.

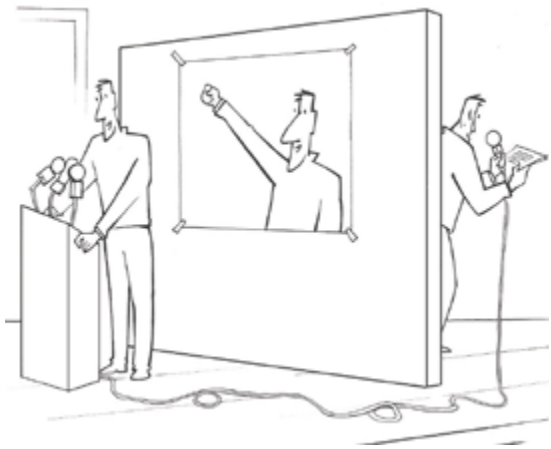
Durch die sehr vereinfachte Darstellung lassen sich Wireframes mit den entsprechenden Tools von fast jedem erstellen. So könnte in unserem Beispiel Ilona Neuhaus als Architektin, oder auch Lina ihre Vorstellung vom System mittels eines Wireframes visualisieren und vermitteln.

Abbildung 17.3: Wireframe

17.3.2 Funktionaler Prototyp – erlebte Funktion

Ein funktionaler Prototyp ist ein funktionsfähiges Modell, das im Entwicklungsprozess als Demonstrator eingesetzt wird. Funktionale Prototypen simulieren die Funktonalität des Systems und zeigen deren Ergebnisse an der Benutzungsschnittstelle. Sie sind eine Middle- bis High-fidelity-Abbildung des zukünftigen Systems, wobei das Aussehen dem Endprodukt nicht völlig gleichen muss, da hier der Schwerpunkt nicht auf dem Oberflächendesign liegt. Funktionale Prototypen ermöglichen es den Stakeholdern die Interaktion mit dem späteren System zu erleben, zu verstehen und zu kommentieren. Um den Entwicklungsaufwand zu minimieren, sind sie meist nicht mit dem Backend verbunden und liefern damit keine realen Daten, sondern arbeiten mit hinterlegten Testdatensätzen. Dennoch lässt sich das Systemverhalten bezüglich der Funktionalität erleben und beurteilen. Um Wissen über ein System zu vermitteln, eignen sich funktionale Prototypen hervorragend, da der Prototyp sehr viel Wissen in einer leicht aufnehmbaren Form darstellt. Funktionale Prototypen können iterativ zu realen Systemen weiterentwickelt werden.

17.3.3 Mock-up der Oberfläche – das Designmodell



Mock-ups der Oberflächen dienen zur Vermittlung der visuellen Seite Ihres Systems und produzieren dadurch Feedback. Ein Mock-up repräsentiert das Design einer Oberfläche. Es ist ein Middle- oder High-fidelity-Oberflächenprototyp. Mock-ups werden häufig auf der Grundlage von Wireframes erstellt und detaillieren diese. Stellen Sie sich ein Mock-up wie einen Screenshot der späteren Oberfläche des Produkts vor. In ihm werden unter anderem bereits die verwendete Typografie und das Farbschema abgebildet. Das Ziel ist es, so genau wie möglich das spätere Erscheinungsbild der zu erstellenden Oberfläche abzubilden. Mock-ups sind somit Bilder zur Wissensvermittlung und damit treffen die Aussagen aus Abschnitt 17.4 „Bilder zur Vermittlung von Wissen“ zu.

Ein Mock-up wird entweder mit einem Grafikprogramm oder einem spezialisierten Tool erstellt und der Erstellungsprozess ist viel aufwendiger als der eines Wireframes. Nach ihrer Verwendung zur Ermittlung oder Vermittlung von Wissen landen Mock-ups in der Spezifikation oder werden iterativ zur Oberfläche des Produkts erweitert.

17.3.4 Gute Prototypen für eine gute Vermittlung

Prototyping ist eine sehr effektive Möglichkeit, um Wissen zu vermitteln. Da die Entwicklung eines guten Prototyps aufwendig ist, sollten Sie sich der Vor- und Nachteile des Prototypings bewusst sein.

- Keine Notationskenntnisse erforderlich
- Gerade funktionale Prototypen regen zum Spielen an und damit zu einer intensiven Auseinandersetzung mit der Funktionalität.
- Grafische Informationen (z. B. Aussehen einer Oberfläche) können besser als Bild vermittelt werden.
- Üblicherweise universell in allen Sprachen einsetzbar
- Gut mit anderen Vermittlungstechniken kombinierbar

- Interpretierbar
- Schlecht wartbar, da Einzelelemente eines Bildes schlecht attribuierbar/referenzierbar sind und damit Änderungen im Detail schlecht verfolgt werden können.
- Schlecht erkennbar, welche Bestandteile des Prototyps Anforderungen sind und welche nicht

17.4 Bilder zur Vermittlung von Wissen

Ein Bild sagt mehr als tausend Worte .... aber wollen Sie das wirklich alles sagen? 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.

Abbildung 17.4 zeigt die Frontseite des Hauses der Familie Meyer mit der groben Position der Haustür und des geplanten Vordachs. In der Abbildung stimmen weder die exakten Maße des Hauses noch sind die restlichen Elemente wie Fenster, Tür an der exakt richtigen Position. Dieser erste bildliche Entwurf eignet sich dennoch hervorragend als Grundlage für ein Gespräch über die Positionierung von Überwachungskameras (durch rote Punkte gekennzeichnet) und damit zur Vermittlung von Anforderungen und Ideen, was alles auf welche Art überwacht werden soll.

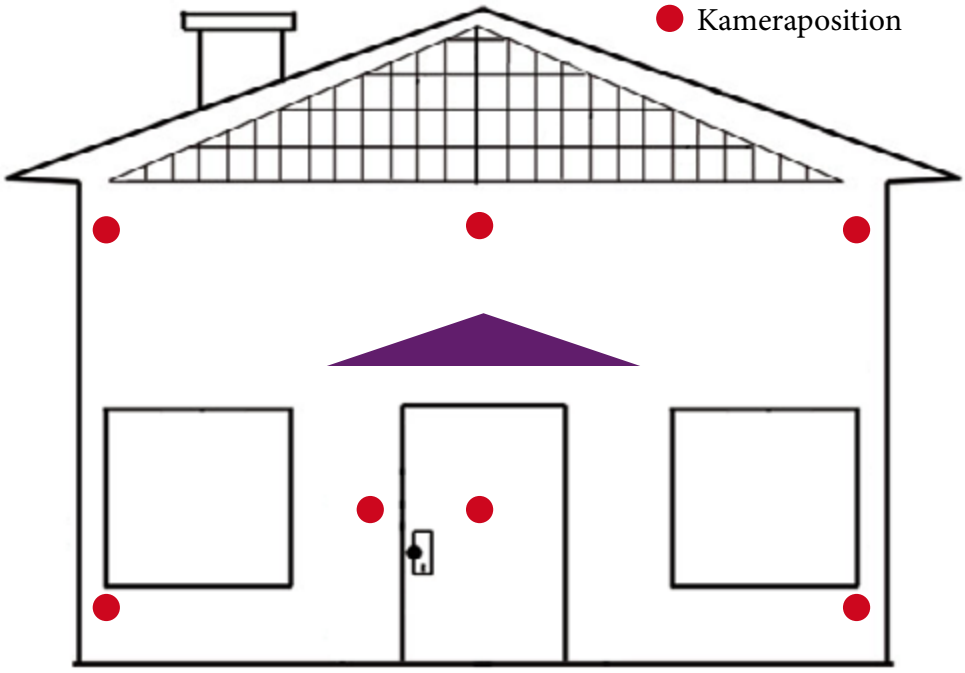


Abbildung 17.4: Die Frontseite des Hauses der Familie Meyer mit der Position der Haustür und des geplanten Vordachs sowie potenziellen Kamerapositionen

Ein Bild hat die Stärke, räumliche Zusammenhänge (also z. B. wo befindet sich die Kamera relativ zum Vordach, das die Aufnahme verhindern könnte?) darzustellen.

Zudem stellt ein derartiges Bild immer eine konkrete Situation dar, wohingegen ein semiformales oder formales Modell abstrakt ist. Das Bild zur Diskussion der Kameraposition zeigt ein ganz spezielles, konkretes Haus, über das diskutiert wird, denn es lässt sich kein generisches Haus zeichnen. Anhand dieses konkreten Beispiels lassen sich manche Fehler finden, die wir im Modell nicht finden (und natürlich auch umgekehrt).

17.4.1 Definition der eigenen Bildsprache

Bei informellen visuellen Darstellungen braucht keiner der beteiligten Personen Notationskenntnisse. Vielmehr erschafft man sich als Team beim Erstellen und in der Diskussion seine eigene Notation, die im Team verstanden wird – oder eben erklärt werden muss (z. B. erläutert die Legende, dass die roten Punkte die möglichen Positionen für die Anbringung einer Überwachungskamera darstellen).

Überlegen Sie sich bei einer informellen Darstellung immer, was selbstverständlich ist – also auch ohne Festlegung oder Erläuterung verstanden wird (hier z. B. dass das größere blaue Rechteck die Haustür darstellt). Erläutern oder definieren Sie, was nicht selbstverständlich ist (z. B. dass die roten Punkte die potenziellen Anbringungspunkte für eine Überwachungskamera darstellen). Dabei müssen Sie Ihr anvisiertes Zielpublikum einschätzen können, mit Rückfragen rechnen oder beim Erläutern Ihres Bildes an der Nasenspitze ablesen, welche Informationen Sie erfolgreich transportiert haben.

17.4.2 Verbindliches von nicht Verbindlichem trennen

Legen Sie bei einer informellen Darstellung fest, was in einem Bild als Anforderung verbindlich ist.

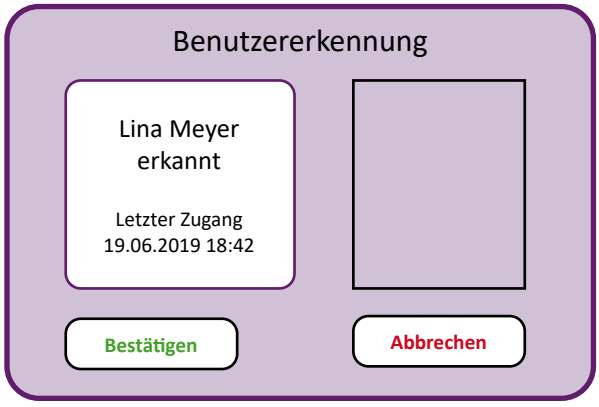


Abbildung 17.5: Beispielhafter Inhalt, der auf dem Zugangsterminal angezeigt wird

Wollen Sie mit der Abbildung 17.5 festlegen, dass es zwei Schaltflächen gibt, die den Prozess bestätigen oder abbrechen? Oder sind damit die Begriffe Bestätigen und Abbrechen festgelegt? Ist damit geklärt, dass Bestätigen grün und Abbrechen rot sein muss? In genau diesem Farbton? Müssen die Schaltflächen genau diese Form und Größe haben? Müssen sie sich an genau dieser Position befinden? Sind die abgerundeten Ecken Pflicht? Muss Bestätigen immer links von Abbrechen stehen? Was steht bei „Letzter Zugang", wenn die Person noch nie Zugang erlangte (nichts oder eine Info, dass die Person noch nie Zugang hatte)?

Ein Bild sagt mehr als tausend Worte – aber welche davon meinen Sie als Anforderung ernst?

Definieren Sie, ob das Bild

- farbverbindlich ist (Fordern Sie genau diesen Farbton?),
- positionsverbindlich ist (Muss das Element pixelgenau an dieser Stelle sein?),
- größenverbindlich ist (Stimmt die Größe des Objekts?),
- textverbindlich ist (Muss genau dieser Text dort stehen oder ist das ein Platzhalter? Bei der Überschrift Zugangsberechtigung könnten Sie den Text verbindlich meinen, beim Datum des letzten Zugangs hätten Sie vermutlich gerne immer das Datum des letzten Zugangs und nicht den 19.06.).

Diese Festlegungen sollten Sie mit Ihrem Bild kommunizieren, damit Ihre RezipientInnen von den gleichen Annahmen ausgehen. Dieses Vorgehen empfehlen wir Ihnen auch beim Einsatz von Prototypen, obgleich diese nicht ganz so informell wie Bilder sind. Nähere Informationen zu den Definitionen finden Sie unter www.sophist.de/re7/kapitel17. Ein Beispiel dazu finden Sie in Abschnitt 13.5 „Steckbrief Anforderungen an die Benutzungsoberfläche“.

17.4.3 Kombination Bild mit anderen Techniken der Wissensvermittlung

In der Praxis werden Bilder oftmals mit anderen Techniken der Wissensvermittlung kombiniert. Hier folgen ein paar Spielarten, die wir häufig antreffen.

Ein Bild dient als Erzählgrundlage. Die wissende Person malt, während sie ihr Wissen ausspricht, ein Bild und nutzt dieses, um ihre Erzählung zu strukturieren oder die Aufmerksamkeit der KollegInnen durch Zeigen auf bestimmte Bildbereiche zu lenken.

- Die wissende Person malt ein Bild und lässt sich von den RezipientInnen erklären, was sie dort an Informationen sehen. Falls die Informationen, die sie vermitteln will, noch nicht eindeutig vermittelt werden, bessert sie im Bild nach, bis die Information ankommt.
- Die wissende Person malt ein Bild, das einen ersten Eindruck vermittelt. Alle Informationen, die sie für wichtig hält, expliziert sie schriftlich und fügt diese dem Bild hinzu. Das Bild dient hier zur groben Orientierung und liefert den Kontext für die zugehörigen Anforderungen – hat aber den Charakter eines Kommentars, der juristisch nicht verbindlich ist.
- Die wissende Person nutzt sowohl ein Bild als auch Text, um ihre Anforderungen zu vermitteln. Achten Sie in diesem Fall darauf Redundanzen und Widersprüche

zu vermeiden. Aspekte, die besser im Bild zu vermitteln sind (z. B. Positionierung relativ zueinander) sollten dort dargestellt werden. Aspekte wie geforderte Schriftgröße/Schriftart sollten lieber textuell gefordert werde, auch wenn Schriften im Bild sichtbar sind.

17.4.4 Bilder für eine gute Vermittlung

Da sehr viele Menschen visuell veranlagt sind und Bilder einen guten Wiedererkennungswert besitzen und gut memoriert werden können, sind sie ein Mittel der Wissensvermittlung, das häufig eingesetzt wird. Bedenken Sie vor der Verwendung allerdings die Vor- und Nachteile dieser Technik.

- **Keine Notationskenntnisse erforderlich**
- **Bestimmte Informationen können besser als Bild vermittelt werden.**
- **Üblicherweise universell in allen Sprachen einsetzbar**
- **Gut mit anderen Vermittlungstechniken kombinierbar**

- **Interpretierbar**
- **Schlecht wartbar, da Einzelelemente eines Bildes schlecht attribuierbar/referenzierbar sind und damit Änderungen im Detail schlecht verfolgt werden können**
- **Schlecht erkennbar, welche Bestandteile des Bildes verbindliche Anforderungen darstellen und welche nicht**

17.5 Gemeinsam Artefakte erstellen

Eine weitere Möglichkeit Anforderungen zu vermitteln ist das gemeinsame Erstellen zusätzlicher Artefakte. Besonders gut geeignet ist dabei das gemeinsame Erstellen von Testfällen zu den Anforderungen. Es ist aber auch denkbar, dass andere Artefakte, zum Beispiel Architektur- oder Designdokumente oder Bedienungsanleitungen, erstellt werden. Gerade wenn Sie im Systems-Engineering tätig sind, könnte die Erstellung von Architekturdokumenten sinnvoll sein (siehe Kapitel 23 „Systems-Engineering“). Suchen Sie ich Artefakte aus, die sowieso erstellt werden müssen, um nicht am Ende das erstellte Artefakt wieder wegzuwerfen.

17.5.1 Vorbereitung

Bevor Sie mit dem gemeinsamen Erstellen der Artefakte beginnen, sollten Sie ein paar Dinge festlegen. Zunächst sollten Sie sich überlegen, welche Artefakte erstellt werden sollen. Dafür kann es auch sinnvoll sein sich die Meinungen anderer Personen darüber, welche Artefakte geeignet sind, einzuholen. Wir werden in diesem Kapitel speziell auf das Erstellen von Testfällen eingehen, daher sprechen wir ab jetzt nur noch von diesen.

Weiterhin sollten Sie den inhaltlichen Umfang festlegen. Das bedeutet: Zu welchem Thema wollen Sie die Testfälle erstellen? Das kann zum Beispiel die Auswahl ein oder mehrerer User-Storys sein. Es ist ratsam den Umfang nicht zu groß zu wählen, da Sie sonst beim Erstellen der Testfälle leicht den Überblick verlieren. Planen Sie lieber ein paar mehr aber kürzere Termine ein.

Sammeln Sie alle Informationen, die Sie zu dem festgelegten Scope haben, denn Sie sollten darauf vorbereitet sein, alle möglichen Fragen gestellt zu bekommen. Überlegen Sie sich dazu auch, welche Visualisierungsmöglichkeiten Sie nutzen können. Gerade wenn der Scope etwas größer ist und die Arbeitssession dementsprechend länger dauern kann, ist es hilfreich Informationen zu visualisieren, um sie für die Zeit des gemeinsamen Arbeitens präsent zu halten.

Letztendlich sollten Sie sich noch bewusst machen, wer die Beteiligten der Arbeitssession sein sollen. Das heißt: Wer soll eigentlich die Informationen zu den Anforderungen erhalten? Bedenken Sie hierbei, dass vor allem die Personen, die aktiv mitarbeiten, am meisten von der Vermittlung der Anforderungen profitieren. Dementsprechend sind größere Gruppen weniger geeignet, wenn alle Beteiligten viel detailliertes Wissen erhalten sollen.

17.5.2 Überblick geben

Sobald die Vorbereitungen abgeschlossen sind, setzen sich alle Beteiligten zusammen, um gemeinsam die Testfälle zu erstellen. Es empfiehlt sich hierbei, an einem Ort zusammenzukommen. Sicherlich ist dies nicht immer möglich und Sie werden auf andere Mittel wie Webmeetings ausweichen müssen. Die Erfahrung hat aber gezeigt, dass das Nebeneinandersitzen am besten geeignet ist.

Zunächst gibt die vermittelnde Person einen Überblick über die Anforderungen, welche vermittelt werden sollen. Dabei sollen noch keine Details besprochen werden, sondern alle Beteiligten sollen erst mal den Gegenstand der

Vermittlung kennenlernen. Falls nötig, können Sie hierfür eine moderierende Person hinzuziehen, falls die Beteiligten immer wieder in Detaildiskussionen abzuschweifen drohen.

Hilfreich für das Verständnis und den Einstieg ist es zu diesem Zeitpunkt, den Beteiligten auch Hintergrundinformationen zu den Anforderungen zu liefern, wie zum Beispiel eine Begründung für die Anforderungen. Auch sollte zu diesem Zeitpunkt der Kontext, in dem man sich mit den Anforderungen bewegt (nicht zu verwechseln mit dem Systemkontext), klar herausgestellt werden. Als Einstieg empfehlen wir Ihnen Storytelling (siehe Abschnitt 17.1 „Storytelling“) zu verwenden. Unterstützend sollten Sie von diversen Visualisierungsmöglichkeiten Gebrauch machen. Das kann von einfachen Stichpunktlisten über kleine Zeichnungen bis hin zu skizzierten Aktivitätsdiagrammen reichen.

17.5.3 Erstellen der Testfälle

Nachdem die Beteiligten einen Überblick erhalten haben, wird gemeinsam der erste Testfall erstellt. Dazu sollten Sie zunächst den Normalfall betrachten, zum Beispiel: Eine autorisierte Person kommt zur Wohnungstür und diese entriegelt sich automatisch. Der erste Testfall behandelt den Fall, dass die Person autorisiert ist, dass die Tür verriegelt ist und entriegelt werden kann etc. Es werden also keine Ausnahmen oder Sonderfälle betrachtet.

Nachdem dieser Testfall beschrieben ist, werden Schritt für Schritt die Ausnahmen und Sonderfälle betrachtet. Hierzu kann das *RE*gelwerk in Kapitel 9 „Das SOPHIST-*RE*gelwerk“ eine gute Hilfe sein, um herauszufinden, ob sämtliche Fälle berücksichtigt worden sind. Hierbei sollten Sie konkrete Beispiele nutzen. Anstatt den Fall zu betrachten, dass Teile vom Gesicht der Person bedeckt sind, sollten Sie beispielsweise besser darüber sprechen, was in den Fällen passieren soll, wenn die Person einen Schal oder eine Mütze trägt. Diese konkreten Beispiele erleichtern das Auffinden sämtlicher Sonderfälle gegenüber einer abstrakten Betrachtung des Sachverhalts.

Wer im Einzelnen die Testfälle schreibt, ist egal. Das kann die testende Person sein, da es sich um ihre Domäne handelt, es ist aber auch denkbar, dass eine der anderen beteiligten Personen das macht. Überlegen Sie sich, ob Sie nicht einen Wechsel der schreibenden Person einführen können, sodass jeder in den Genuss kommt Testfälle zu schreiben.

Für die Formulierung der Testfälle bieten sich verschiedene Techniken an. Hier eignen sich das Given-When-Then-Template (siehe Abschnitt 19.6.2 „Aufbau und Inhalt von Akzeptanzkriterien für User-Storys“) oder aber auch andere Formulierungstemplates für Testfälle. Sie sollten auf jeden Fall darauf achten, dass eine Formulierung gewählt wird, die für alle Beteiligten verständlich ist. Denn alle Beteiligten sollen sich am Ende einig sein, was die gewünschten Anforderungen konkret sind. Weitere Informationen zum Schreiben von Testfällen finden Sie in Abschnitt 14.5 „Testfälle“.

17.5.4 Gemeinsam erstellte Artefakte für eine gute Vermittlung

Die gemeinsame Erstellung von Artefakten ist eine sehr effektive Möglichkeit, um Wissen zu vermitteln. Da die Entwicklung der Artefakte aufwendig ist, sollten Sie sich der Vor- und Nachteile bewusst sein.

- **Gemeinsames Bearbeiten der Anforderungen stärkt das gemeinsame Verständnis.**
- **Es werden Artefakte erstellt, die für weitere Arbeiten benötigt werden.**
- **Durch das gemeinsame Arbeiten erhöhen sich die Chancen, dass alle Sonderfälle berücksichtigt wurden.**

- **Es wird vorausgesetzt, dass alle Beteiligten gemeinsam arbeiten können.**
- **In manchen Szenarien zu aufwendig und schießt über das Ziel hinaus (zum Beispiel Szenario 1 „Kundenanfrage bearbeiten“); möglicherweise ist dazu ein anderes Zielartefakt hilfreich.**

Index

SYMBOLE

A

B

C

D

E

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z