Teil I

Einführung

- **Kapitel 1:**
  In medias RE – Grundlegendes zum Requirements-Engineering
- **Kapitel 2:**
  Die Meyers und ihr Traum vom Smart Home.
- **Kapitel 3:**
  Requirements-Engineering im Überblick – Von der Idee zur Anforderung
- **Kapitel 4:**
  RE ist nicht gleich RE – das richtige Maß finden

Chris Rupp

1

In medias RE – Grundlegendes zum Requirements-Engineering

1.1 Motivation für ein erfolgreiches Requirements-Engineering

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

Damit all das zu einem Traum für die Menschheit wird und nicht im Albtraum endet, ist es wichtig, dass die Systeme das tun, was sie tun sollen und was wir von ihnen wollen. Genau hier setzt Requirements-Engineering an. Diese immer komplexeren Leistungen, die Systeme übernehmen, müssen erfunden oder ermittelt, analysiert und vermittelt werden und das entstandene Wissen (die Anforderungen) muss dann oft auch dokumentiert und verwaltet werden. Vorher müssen Sie die vom System zu unterstützenden Geschäftsprozesse verstehen und skizzieren.

Die Kunst dabei ist es, auf die mannigfaltig vorliegenden Rahmenbedingungen einzugehen. Requirements-Engineering bedeutet hier die richtige Form für den richtigen Zweck zu finden. Wir haben uns für dieses Buch auf ein Beispiel – unser Smart Home – geeinigt und drei typische Szenarien beschrieben, in denen Requirements-Engineering in der Realität häufig stattfindet. Das Beispiel finden Sie in Kapitel 2 „Die Meyers und ihr Traum vom Smart Home“, die Szenarien in Kapitel 4 „RE ist nicht gleich RE“.

Und hier noch eine ganz persönliche Geschichte, die Sie zu unserem durchgehenden Beispiel in diesem Buch hinführt. Es handelt sich dabei um ein Beispiel für Storytelling, eine Vermittlungstechnik, die sie in Kapitel 17 „Storytelling, User-Storys & Co.“ kennenlernen werden. Ganz nebenbei erläutert diese Geschichte, warum Requirements-Engineering wichtig ist.

Ich lebe auf dem Land, und zwar wirklich auf dem Land – auf einem Bauernhof in einem abgelegenen Weiler. Hier guckt nicht ständig ein Nachbar über den Zaun, ob bei mir alles o. k. ist. So kam mir der Gedanke, dass etwas mehr Sicherheit und gerne auch Komfort durch mehr Digitalisierung meines Anwesens eine gute Idee wäre. Super wäre dann, wenn die zusätzliche Intelligenz im Haus auch noch für eine ökologische Nutzung der durch die vorhandene Solar- und Fotovoltaik-Anlage erzeugten Energie sorgen könnte. Wichtig waren mir aber erst mal ein paar Kameras und eine neue Haustüre mit sinnigen Features für Sicherheit und Komfort sowie eine Alarmanlage.

Auf die Idee kam ich, da gerade jeder in meinem Bekanntenkreis sein Haus smart machte, irgendwie begeistert davon erzählte und mir auf dem Smartphone zeigte, was gerade bei ihm so im Garten abging (meist nix). Da würde bei mir daheim schon mehr abgehen, denn ich habe zwei große Hunde und einige Katzen – und da würde mich interessieren, was die tagsüber so treiben.

Natürlich hatte ich überhaupt keine große Lust aus der Smartifizierung ein großes Ding zu machen.

Meine erste Vision war ein Standardprodukt von meinem Elektriker installieren zu lassen – und initial den Fokus nur auf die Eingangstüre und die Kameras zu legen. Da ich im ersten Stock schlafe, hätte ich gerne noch mittels einer Alarmanlage dafür gesorgt, dass ich rechtzeitig geweckt werde, wenn jemand gerade einbricht.

Auf die Idee Standardprodukt kam ich, nachdem ich mit einem Kollegen gesprochen hatte, der ein IT-Freak ist. Die Komplexität und der Ideenreichtum seines selbst konfigurierten Systems haben mich derart überfordert – ich fühlte mich nach einigen Minuten schon durch die Erzählung abgeschreckt. Somit besuchte ich einen weniger freakigen Kollegen, der mit seiner Frau und den Kindern in der Gegend wohnt. Sein System war wirklich Standard – vom Hauselektroniker eingebaut. Im Großen und Ganzen bestand es aus ein paar Kameras im Außenbereich, deren Bilder er über sein Handy ansehen konnte, und aus einer Alarmanlage für das Erdgeschoss, die er scharf schaltete, wenn er das Haus verließ oder wenn er zum Schlafen mit der Familie in das obere Stockwerk ging. Bei dem Besuch gingen wir dann alle nach oben und er zeigte mir, wie einfach es war, über das Smartphone jetzt die Alarmanlage zu aktivieren. Wir checkten auch gleich mal, was man am Abend noch so über die Außenkameras sehen kann – und das war leider etwas enttäuschend, denn die Nachtsicht war damals bei der Einrichtung anscheinend keine Anforderung gewesen. Mein Kollege erzählte, dass er aber ein Bild auf das Handy gesendet bekäme, sobald die Kamera eine Bewegung detektieren sollte. Bei der Idee wurde mir angst, denn in meinem Garten springen nachts Rehe, Katzen, Füchse und sonst noch so einiges herum. Das würde sehr viele Meldungen und Bilder auf meinem Handy bedeuten. Als wir nun alle so über das Smartphone gebeugt dastanden, ging plötzlich die Alarmanlage los. Irgendwie musste das System wohl doch nicht wie gewünscht funktionieren. Sofort fiel mir auf, dass Pirata und Nemo – meine beiden Hunde – nicht mehr neben uns waren. Sie hatten anscheinend die Erzählung langweilig gefunden und die Hauserkundung im unteren Stockwerk fortgesetzt – überwacht durch die Alarmanlage, die sie sofort für Einbrecher hielt. Frust machte sich langsam bei mir breit, denn anscheinend bin ich doch kein Standardkunde – nachts schlafen meine Hunde im Gang im Erdgeschoss und die Katzen kommen und gehen, wann sie wollen. Da wir dann schon mal bei den Problemen einer derartigen Anlage angekommen waren, packte mein Kollege gleich mal aus, was so alles nicht geht in dem Standard-Ding. Das Zufügen und Löschen weiterer Personen, die mittels Key Zutritt zum Haus haben, war sperrig und man konnte lediglich Zutritt erteilen oder nicht – also nix mit Profilen, wie „der Handwerker darf nur an einem bestimmten Tag zu einer gewissen Zeit rein“. Und eine offene Frage quälte meinen Kollegen: Was passiert, wenn es mal im Haus brennt und die Kinder noch drin sind. Man konnte mit dem System nicht Fernentriegeln – sodass die Feuerwehr bei der einbruchsicheren Türe und den sehr massiven Fenstern nie ins Haus käme, um die Kinder zu retten ...

Mir wurde klar: Wenn ich mich nur zehn Minuten lang hinsetzen würde, dann würden mir sicherlich einige Anforderungen einfallen, die nicht Standard sind, bei meiner Art zu leben aber unabdingbar. Mir war aber auch klar, dass mein Wissen zum Thema nicht ausreicht, um zu überblicken, was ich noch so alles beachten und fordern sollte. Ich bin eben Spezialist bezüglich meiner Lebensumstände, aber kein Smart-Home-Spezialist. Und klar – typisch, dass ich als Requirements-Engineer erst mal x Versuche unternehme dem Problem einer sinnvollen Analyse und sauberer Anforderungen zu entgehen –, der Schuster hat ja angeblich auch die schlechtesten Schuhe ...

Meine Rettung ist wohl doch nur ein etwas ausführlicheres Requirements-Engineering meiner Wünsche und Lebensgewohnheiten, um dann bei den richtigen Anforderungen zu enden. Damit ich weiß, was ich mir noch so alles wünschen kann, habe ich jetzt Kontakt zu einer Architektin und einem Ansprechpartner eines Smart-Home-Anbieters aufgenommen – und so habe ich vermutlich eine realistische Chance auch bei einem System zu enden, das für mich Sinn ergibt.

Der Requirements-Engineer – Mittler zwischen den Welten

Konzentrieren wir uns nun auf die Person, die das Requirements-Engineering durchführen soll. Der Requirements-Engineer (auch als Systemanalytiker, Anforderungsanalytiker oder Business Analyst bezeichnet) ist ein Mittler zwischen den Welten. In der agilen Entwicklung übernimmt diesePerson eine Mittlerrolle – und damit gelten für ihn all die Anforderungen, die wir uns hier für den Requirements-Engineer wünschen. Oftmals wird ein Product-Owner durch eine weitere Rolle – den Product-Owner-Support – unterstützt, für den dann natürlich ebenfalls die folgenden Anforderungen gelten. Wenn Sie in Ihrer Entwicklung (egal ob agil oder eher klassisch) eine Rolle wahrnehmen, die die Bedürfnisse von KundInnen, BenutzerInnen, KäuferInnen ... an die Menschen weiterleitet, die das Produkt designen, erstellen ..., dann sollten Sie sich im Folgenden durch die Bezeichnung Requirements-Engineer angesprochen fühlen.

Als Requirements-Engineer interagieren Sie mit vielen Menschen, vor allem mit den späteren Anwendern des Systems, aber auch mit den SystemarchitektInnen, den EntwicklerInnen, dem Testteam und dem Projektmanagement, und haben großen fachlichen Einfluss auf die Systementwicklung. Ein Requirements-Engineer erhebt und dokumentiert die Wünsche und Anforderungen der Stakeholder an das System oder Produkt, moderiert und vermittelt zwischen Stakeholdern und allen Beteiligten, arbeitet als Katalysator für Entscheidungen der Stakeholder und muss daher großes Fingerspitzengefühl für die Bedürfnisse der Beteiligten besitzen.

Die Rolle des Requirements-Engineers muss nicht immer von einer Person mit Informatikausbildung wahrgenommen werden, um die Bedürfnisse aller Stakeholder erfolgreich ermitteln, analysieren und abgleichen zu können. Es ist also nicht ungewöhnlich, dass neben Entwicklern auch Mitarbeitende des Fachbereichs diese Rolle besetzen.

Wichtig ist nicht die berufliche Herkunft der Person, sondern es sind ihre persönlichen fachlichen und methodischen Kompetenzen, die zählen.

1.1.1 Anforderungen an einen Requirements-Engineer

Für eine derartige Tausendsassa-Rolle braucht man gewisse Eigenschaften oder muss in sie hineinwachsen. Das IREB e.V. (International Requirements Engineering Board e.V.) hält für einen Requirements-Engineer neben der fachlichen und methodischen Kompetenz die folgenden Eigenschaften [CPRE11 und CPRE19] für relevant:

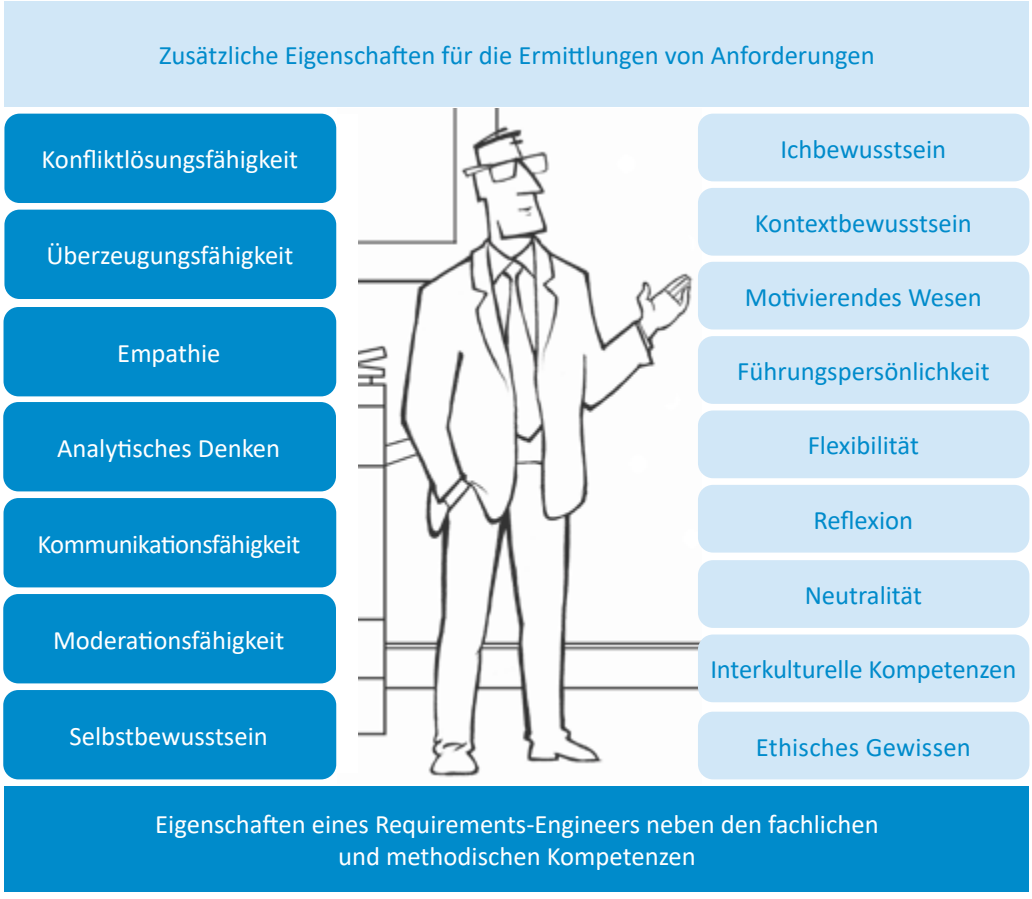


Abbildung 1.1: Eigenschaften eines Requirements-Engineers laut IREB

Mit den vorgestellten Eigenschaften (die einen werden bei Ihnen etwas mehr ausgeprägt sein, andere etwas weniger) und Ihrer fachlichen Kompetenz sind Sie gut ausgestattet. Mit diesem Buch helfen wir Ihnen jetzt noch, Ihre methodischen Kompetenzen zu perfektionieren – damit Sie in Zukunft auf dem Projektparkett glänzen können.

1.1.2 Der Wachstumsprozess eines Requirements-Engineers

Requirements-Engineering ist ein großes Übungsfeld, in dem Sie lernen und immer besser werden können. Für Ihr persönliches Wachstum sollten Sie Ihre eigenen Fähigkeiten und Eigenschaften immer wieder selbst oder im Team reflektieren. Dazu eignen sich Retrospektiven oder Feedbackrituale. Stellen Sie sich regelmäßig die folgenden Fragen:

- Habe ich bei der Auswahl einer Methode oder Notation oder eines Tools die richtige Wahl getroffen? Wie bewusst war diese Wahl? Lieferten die eingesetzten Methoden den erwarteten Erfolg? Standen dabei Aufwand und Nutzen in einem guten Verhältnis?
- War ich überzeugend/empathisch/kommunikativ gegenüber dem Team? Konnte ich Konflikte frühzeitig erkennen und lösen? Habe ich mit meiner Moderation das Team unterstützt?
- Habe ich ausreichend Neutralität bewahrt? Wie steht es mit meiner interkulturellen Kompetenz?
- Hätte ich etwas anders machen können? Was habe ich dabei gelernt?

Geeignet sind natürlich auch die typischen Retrofragen: Was war gut? Was war schlecht? Was behalten wir? Was ändern wir?

Fokussieren Sie sich bei Ihrem eigenen Entwicklungsprozess immer auf wenige Punkte, gerne auch auf Details, z. B. nur auf Ihre Fähigkeit in einem Interview zuzuhören. So erzielen Sie schnell messbare Erfolge. Nutzen Sie dabei unterschiedliche Feedbackgelegenheiten wie KollegInnen, die Sie beobachten, Feedbacks von KundInnen, Video- oder Audioaufzeichnungen etc. Falls Sie das Gefühl haben, Ihnen fehlt Wissen und Können, dann sind Mentoring oder der Besuch einer Schulung die geeigneten ersten Schritte. Sollten Sie schon viel Erfahrung haben, so ist es meist am lehrreichsten, ebenfalls erfahrenen KollegInnen beim Arbeiten über die Schulter zu sehen.

Es ist ein langer Weg, ein guter Requirements-Engineer zu werden, den es sich aber zu gehen lohnt. Durch viel Erfahrung und Reflexion wachsen die Fähigkeiten, ändert sich die Einstellung und entwickelt sich die Persönlichkeit. Lernen ist ein Prozess, zu dessen Beginn alle unbedarfte AnfängerInnen sind und zu dessen Ende hoffentlich alle zu ExpertInnen geworden sind. Diese Entwicklung wurde von Hubert und Stuart Dreyfus [Dreyfus00] im „Dreyfus Model of Skill Acquisition“ in fünf Stufen aufgeteilt (siehe Abbildung 1.2). Wenn Sie wissen, auf welcher Stufe Sie oder Ihre Teammitglieder stehen, können Sie die geeigneten Maßnahmen ableiten.

AnfängerInnen haben wenig bis gar keine Erfahrung und benötigen klare Regeln zum Vorgehen. Sie sind nicht am Lernprozess interessiert, nur daran, die Aufgabe zu bewältigen, denn sie plagen Versagensängste. Weder selbstständig arbeiten oder Ergebnisse reflektieren noch relevante Informationen selektieren ist möglich.

Fortgeschrittene lösen sich bereits ansatzweise von fixen Regeln und Vorgaben. Sie können Aufgaben allein bewältigen, haben aber noch Schwierigkeiten mit der Lösung von Problemen und anspruchsvollen Sachverhalten. Sie suchen gezielt nach Informa-

tionen, die ihnen wichtig erscheinen, werden aber manchmal wichtige Informationen als irrelevant abtun, da sie noch nicht über genug Wissen und Erfahrung verfügen, um verallgemeinernde Ableitungen und Prinzipien zu formulieren. Ihr größter Vorteil gegenüber den AnfängerInnen ist die Fähigkeit, eigene Vorgehensstrategien zu entwickeln, die sich auf andere Situationen anwenden lassen.

Kompetente sind in der Lage, gedankliche Konzeptmodelle eines Problembereichs zu entwickeln und mit ihnen erfolgreich zu arbeiten. Sie können selbstständig Probleme erkennen und beheben, ohne dass sie diese vorher bewältigt haben müssen. Dazu genügen bisherige Erfahrung und die Fähigkeit zu planen. Sie sind einfallsreich und entschlossen, bringen aber nicht genug Erfahrung mit, um über ihr Vorgehen reflektieren oder sich selbst korrigieren zu können.

Gewandten ist das holistische (ganzheitliche) Konzept eines Wissensbereichs zum Greifen nahe. Sie haben die konzeptuelle Grundstruktur der Materie bereits vor Augen und können sich mühelos selbst verbessern und Handlungsschritte korrigieren. In dieser Stufe findet der entscheidende Prozess des Dreyfus-Modells statt, der sich wie eine ausgrenzende Trennlinie durch die Entwicklungsstufen zieht. Auf Grundlage ihrer langjährigen Erfahrung können die Gewandten über ihr fachliches Handeln reflektieren und ihre eigenen Leistungen abgleichen und bewerten. Die Möglichkeit zur Reflexion befähigt sie außerdem, aus den Erfahrungen anderer zu lernen, ohne auf eigenes gespeichertes Handlungswissen zurückgreifen zu müssen.

ExpertInnen werden gesteuert durch eine uns magisch anmutende Erfahrung und undurchschaubare mentale Modelle. Sie suchen selbstständig nach besseren Wegen und Lösungen auf Basis ihrer Intuition. Diese Intuition ist verantwortlich dafür, dass sie für ihre Entscheidung oftmals keine Erklärung parat haben: „So ist das einfach." ExpertInnen besitzen ein unterbewusstes Gespür dafür, relevante Dinge von irrelevanten zu trennen, und erkennen Probleme und Ungereimtheiten durch den Abgleich bestimmter Muster aus dem Vorrat ihres reichhaltigen Erfahrungsschatzes.

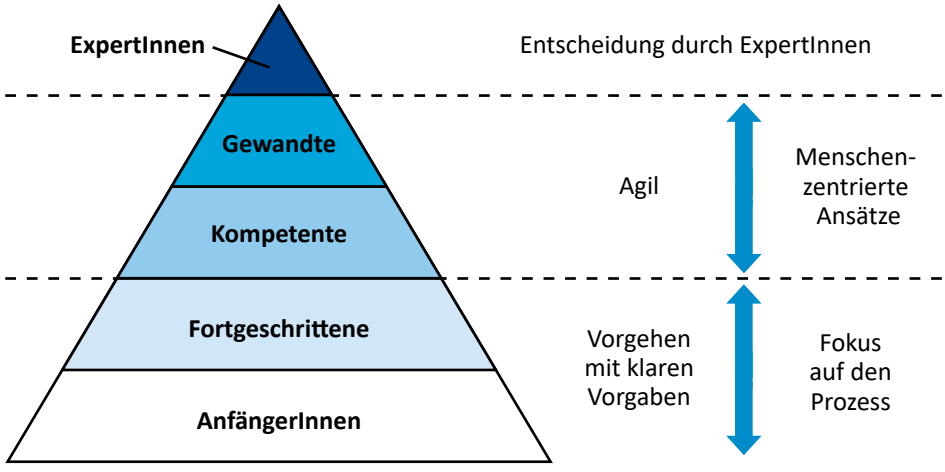


Abbildung 1.2: Wissensstufen und korrespondierende Vorgehensmodelle

Abhängig von der Stufe, auf der sich die Mehrheit der Teammitglieder befindet, sollten Sie das Vorgehensmodell Ihres Entwicklungsvorhabens wählen – sofern Sie die Wahl haben. Mit AnfängerInnen und Fortgeschrittenen erzielen Sie mit Vorgehen, die klare Anweisungen vorgeben, mehr Erfolge. Besteht Ihr Team größtenteils aus Gewandten und Kompetenten, bietet sich Agilität an.

1.2 Das Requirements-Gehirn – die Anforderungssammlung

Viele Vorgehensmodelle schlagen Dutzende oder Hunderte von Ergebnistypen vor, die im Laufe der Systementwicklung erzeugt werden sollen. Agile Entwicklung [Hruschka02] konzentriert sich auf drei maßgebliche Ergebnistypen (Artefakte), die konsequent weiterentwickelt werden. Um von der physischen Form der Ergebnisse abzulenken und nicht über Papierdokumente, Intranetseiten oder Datenbanken sprechen zu müssen, sprechen wir von Gehirnen als Speichermedien. Stellen Sie sich drei spezialisierte Gehirne vor, in denen Wissen jeweils strukturiert festgehalten wird:

- Das Requirements-Gehirn enthält Wissen über die Problemstellung (z. B. Geschäftsprozesse, fachliche Prozesse, Wünsche an das System ...). Es fokussiert auf die Fragestellung „Wie soll das System sein?“. Den Inhalt des Requirements-Gehirns bezeichnen wir auch als Anforderungssammlung.
- Das Architektur-Gehirn ist gefüllt mit Wissen um die Lösung (z. B. Komponentenaufteilung)
- und das Management-Gehirn enthält Wissen über das Projekt (z. B. Pläne und Schätzungen).

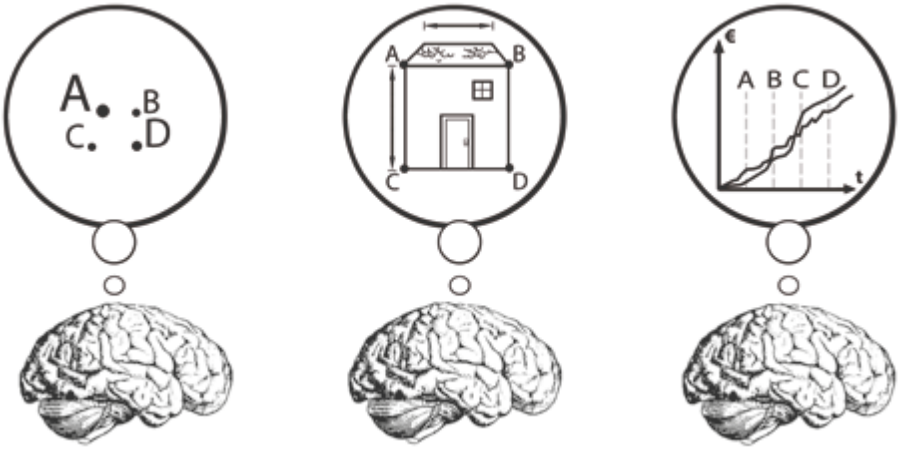


Abbildung 1.3: Die drei Gehirne eines Projektes

Die drei Gehirne sind stark miteinander vernetzt – insbesondere in agilen Entwicklungen, bei denen alle drei Gehirne z. B. in einer User-Story verwoben werden können. Der Inhalt eines Product-Backlogs repräsentiert immer eine Projektsicht und befasst sich mit der Fragestellung „Wie bekomme ich das System realisiert?“ und „Was soll das System leisten?“ Wohingegen eine Anforderungsspezifikation in der Entwicklung die Anforderungssicht repräsentiert und die Fragen „Was soll das System tun?“ und „Welche Eigenschaften soll es haben?“ beantwortet.

Selbstverständlich können wir den Inhalt dieser Gehirne jederzeit auf verschiedene Arten zu Papier bringen, in Präsentationen vermitteln, Videos dazu drehen oder auch mündlich übertragen.

Alles, was wir zu einem Zeitpunkt zu dem jeweiligen Thema wissen, ist im jeweiligen Gehirn gespeichert. Je nach Zustand oder Fortschritt des Entwicklungsvorhabens wird das entsprechende Gehirn mehr oder weniger gefüllt sein. Es gibt kein starr vorgegebenes Schema, was alles im Gehirn festgehalten werden muss. Wir betrachten die Risiken im jeweiligen Bereich und entscheiden danach, wie viel oder wenig wir zu einem Thema festhalten müssen.

In diesem Buch gehen wir speziell auf das Requirements-Gehirn – also die Anforderungssammlung – genauer ein. Dabei sollten Sie allerdings nicht außer Acht lassen, dass zwischen dem Requirements-Gehirn und den beiden anderen Gehirnen ein reger Austausch stattfindet. Kein Gehirn kann für sich alleine in einer Systementwicklung arbeiten!

Die Anforderungssammlung befindet sich im Requirements-Gehirn. Wenn das im Requirements-Gehirn gespeicherte Wissen zu Papier gebracht und ausgedruckt wird, um Informationen für den nächsten Entwicklungsschritt bereitzustellen, dann spricht man von einer Anforderungsspezifikation. Dieses Dokument repräsentiert jeweils einen Stand der bekannten Anforderungen nach einer gewissen Sortierung zu genau einem Zeitpunkt. Ist die Entwicklung des Systems abgeschlossen, so können die Anforderungen natürlich weiterhin verwendet werden, um z. B. das realisierte System zu beschreiben – hier sprechen wir dann von einer Anforderungs- oder Systembeschreibung.

Die Metapher des Gehirns hilft uns, auch in agilen Werten zu denken. Agil durchgeführte Entwicklungen brauchen ebenfalls ein Requirements-Gehirn, das umfangreiches Wissen (die Anforderungssammlung) über die Anforderungen enthält. Der Inhalt des Requirements-Gehirns befindet sich dort im Product-Backlog, wird aber eher über Kommunikation vermittelt als wohlformuliert ausgedruckt.

1.3 Die Disziplin Requirements-Engineering

Was genau ist nun Requirements-Engineering? Welche Aspekte umfasst es? Es ist eine Disziplin rund um Anforderungen, User-Storys, Informationen. Betrachten wir dazu nachfolgend die Definition des IREB.

Definition der Disziplin des Requirements-Engineerings laut IREB [IREB20]

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

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

Dokumentations- und Spezifikationsvorschriften bedeuten übrigens nicht automatisch, dass eine 500-Seiten-Spezifikation erstellt werden muss. In agilen Vorgehen sind eher Product-Backlogs und User-Storys in Verbindung mit Diskussionen über die User-Story-Inhalte angesagt. Wir finden die folgende, von uns SOPHISTen stammende Definition sehr nützlich. Sie hat sich in der Praxis als gut verständlich, umfassend und ausreichend konkret erwiesen und funktioniert für alle Vorgehensweisen von wasserfallartig bis zu extrem agil.

Definition des Begriffs Anforderung nach SOPHIST

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

Wir vermeiden hier bewusst die unserer Meinung nach falsche Aussage, dass Anforderungen dokumentiert sein müssen. Ersetzen Sie den Begriff „dokumentieren“ einfach durch „übermitteln“, denn in manchen Vorgehensmodellen dient die Dokumentation dazu Anforderungen an andere Personen und Teams zu vermitteln. Neben dem Dokumentieren gibt es allerdings auch noch andere Vermittlungstechniken (z. B. Erzählen).

Sagt jemand „Anforderung“, so meint die Person häufig nur einen Teil der Bedeutung, die unsere Definition abdeckt. Sie versteht den Begriff dann als Forderung nach einer Leistung, die „das System“ als ein Endergebnis einer Entwicklung erbringen muss. Dies ist jedoch nur einer der vielen Aspekte des Wortes „Anforderung“. So gibt es beispielsweise Anforderungen in verschiedenen Verfeinerungsgraden, von sehr groben bis hin zu sehr feinen Anforderungen (siehe Kapitel 3 „Requirements-Engineering im Überblick“). Des Weiteren gibt es neben Anforderungen an das System oder das Produkt auch Anforderungen an die Testfälle, Handbücher, Protokolle, Planungsdokumente, den Entwicklungsprozess und so weiter.

Auch die Art, wie eine Anforderung aussehen kann, ist mannigfaltig. Je nachdem, was Ihnen gerade am meisten dient, können Sie eine User-Story, einen Use-Case, eine Story, eine (formalisierte) natürlichsprachliche Anforderung oder auch ein semiformales Modell (eine Darstellung einer Anforderung in Form eines Diagramms) nutzen. Zudem unterscheiden sich natürlich auch die Artefakte, in denen Ihre Anforderungen landen. Es kann der Backlog eines agilen Vorgehens sein, eine Spezifikation oder Anforderungsspezifikation, ein Benutzerhandbuch, ein Ausschreibungsdokument ... oder irgendein anderes Artefakt, das sich eignet die Wünsche an ein System sinnvoll zu beinhalten.

Da schon mehrfach die Begriffe **System** und **Produkt** aufgetaucht sind, wird es Zeit hier unsere Definition anzubieten. In unserem Alltag begegnen uns viele Arten von Systemen. Aber ein Planetensystem, ein Auto, ein Wettersystem oder auch die Software auf unserem Computer haben zwei Dinge gemeinsam:

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

Über diese beiden Eigenschaften wollen wir den Systembegriff definieren, wie wir ihn in diesem Buch verwenden. Daraus folgt, dass wir den Begriff System auf mehreren Ebenen einsetzen können: Das, was für den einen ein Teil seines System ist, ist für den Lieferanten dieses Subsystems sein System.

Demgegenüber steht der Begriff Produkt.

> Unter Produkt verstehen wir etwas, das verkaufbar ist und aus einem System und zusätzlichen Liefergegenständen besteht.

Zusätzliche Liefergegenstände können z. B. ein Wartungshandbuch oder eine Bedienungsanleitung, aber auch ein Servicevertrag sein.

Wir verwenden im Buch den Begriff System – außer wir wollen explizit darauf hinweisen, dass sich eine Aussage speziell auf ein Produkt bezieht.

Nun zu den Haupttätigkeiten des Requirements-Engineerings. In der Literatur finden Sie zu diesem Thema mehrere mögliche Aufteilungen der Tätigkeiten, die Sie im Requirements-Engineering zu leisten haben – natürlich auch mit unterschiedlichen Bezeichnungen. Wir haben uns auf eine Aufteilung und ein Begriffssystem geeinigt, das sich in der Praxis gut merken und vermitteln lässt. Abbildung 1.4 zeigt die SOPHIST-Festlegung der vier Haupttätigkeiten des Requirements-Engineerings: Wissen ermitteln, gute Anforderungen herleiten, Anforderungen vermitteln und Anforderungen verwalten. Auf diese Tätigkeiten werden wir in den folgenden Abschnitten des Buches näher eingehen, um Ihnen dabei auch die verschiedenen Methoden und Techniken vorzustellen.

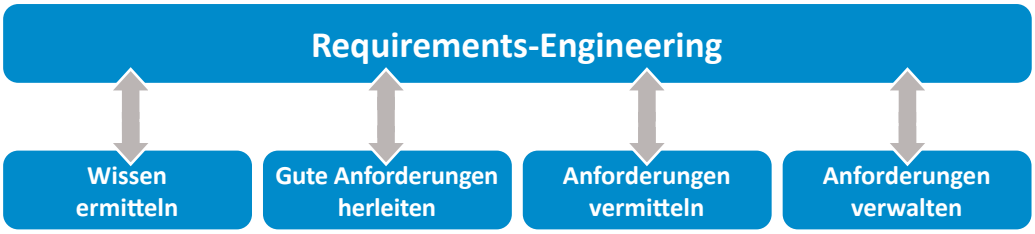


Abbildung 1.4: Die vier Haupttätigkeiten des Requirements-Engineerings

Requirements-Engineering als erster Schritt der Systementwicklung entscheidet maßgeblich über den Erfolg oder Misserfolg der Entwicklung. Verschiedene Untersuchungen zeigen, dass im Rahmen der Systementwicklung die Mehrzahl aller folgenschweren Fehler und Defizite im Requirements-Engineering entstehen. Fehler, die in einem frühen Schritt der Systementwicklung gemacht und erst in darauf aufbauenden Schritten behoben werden (wenn es dafür eigentlich schon zu spät ist), sorgen für einen Schneeballeffekt, bei dem sich die Fehler fortpflanzen und potenzieren. Dieser Effekt wird umso gravierender, je stärker die einzelnen Entwicklungsschritte getrennt sind und je länger die Entwicklungszyklen sind – zwei Aspekte, an denen die agilen Ansätze anknüpfen und Verbesserungspotenziale bieten.

1.4 RE kompensiert die Beschränkung des menschlichen Gehirns

Wären wir Menschen fähig alle Anforderungen im Kopf zu verarbeiten und zu behalten und uns sinnvoll untereinander auszutauschen, dann wäre unser Leben bedeutend einfacher. Das Wissen müsste zwar immer noch an den richtigen Stellen erhoben und mit den relevanten Stakeholdern abgestimmt werden. Man müsste sich auch weiterhin Gedanken über die Wissensvermittlung an die relevanten Beteiligten machen. Aber all die Anforderungen müssten nicht aufwendig dokumentiert und damit haltbar gemacht werden. Leider sind unser menschliches Gehirn und unsere kommunikativen Fähigkeiten aber beschränkt und so hilft uns RE diese Schwächen zu kompensieren.

Das Grundlagenwissen aus diesem Kapitel kann Ihnen helfen die richtige Balance zu finden, wie viel Wissen Sie auf welche Art und zu welchem Zweck dokumentieren. Dass Dokumentation von Anforderungen nicht nur zur Konservierung des Wissens dient, sondern auch andere typische Probleme innerhalb einer Systementwicklung lösen kann, zeigen die folgenden Abschnitte.

1.4.1 Wissen verfällt bzw. diffundiert

Einmal erzeugtes Wissen bleibt nicht unverändert bestehen, sondern geht über die Zeit verloren. Wir sprechen hier von den 3 L's: Lotto, Langeweile und Lastwagen. Ein Lottogewinn (evtl. auch ein höheres Gehaltsangebot eines konkurrierenden Unternehmens oder der Renteneintritt) sorgt genauso dafür, dass Mitarbeitende ihr Unternehmen verlassen, wie Langeweile, die sie zu einem anderen Arbeitgeber treibt. Der Lastwagen als drittes L symbolisiert die Gefahr, dass die Person wegen einer Krankheit oder eines Unfalls ausfällt.

Neben der Fluktuation ist die Vergesslichkeit unseres Gehirns eine wesentliche Ursache für Wissensverlust. Selbst innerhalb kurzer Sprintlängen von nur 14 Tagen gehen bereits große Teile des generierten Wissens verloren.

Belegt wird diese Behauptung durch die Vergessenskurve des Psychologen Hermann Ebbinghaus [Ebb1885, Stangl20], der als Pionier auf dem Gebiet der Gedächtnisforschung gilt. Sie zeigt, wie Wissen über die Zeit diffundiert. In einem Selbstversuch lernte er eine Liste von Silben auswendig, bis er diese zweimal fehlerfrei wiedergeben konnte. In verschiedenen Abständen wiederholte er diesen Versuch. Es zeigte sich, dass bereits nach etwa einer halben Stunde 50 Prozent des initialen Aufwands zum erneuten Lernen notwendig waren. Diskussionswürdig sind lediglich die Informationen, die Ebbinghaus für diesen Versuch verwendete. Es handelte sich um wahllos zusammengesetzte Silben.

Weitere Versuche beispielsweise von Bahrick [Stangl20] anhand von spanischen Vokabeln oder auch von Michel und Novak [Mich90] am Beispiel von Gesetzmäßigkeiten, Gedichten und Prosatexten konnten die Kernaussage von Ebbinghaus belegen. Sie zeigten lediglich, dass die Kurve bei sinnvollen Inhalten in abgeschwächter Form verläuft. Da das Vergessen abhängig vom Stoff und dessen Aufbereitung ist, empfiehlt es sich fundiert darüber nachzudenken, in welcher Form man Wissen vermittelt (Kapitel 17 „Storytelling, User-Storys & Co.“).

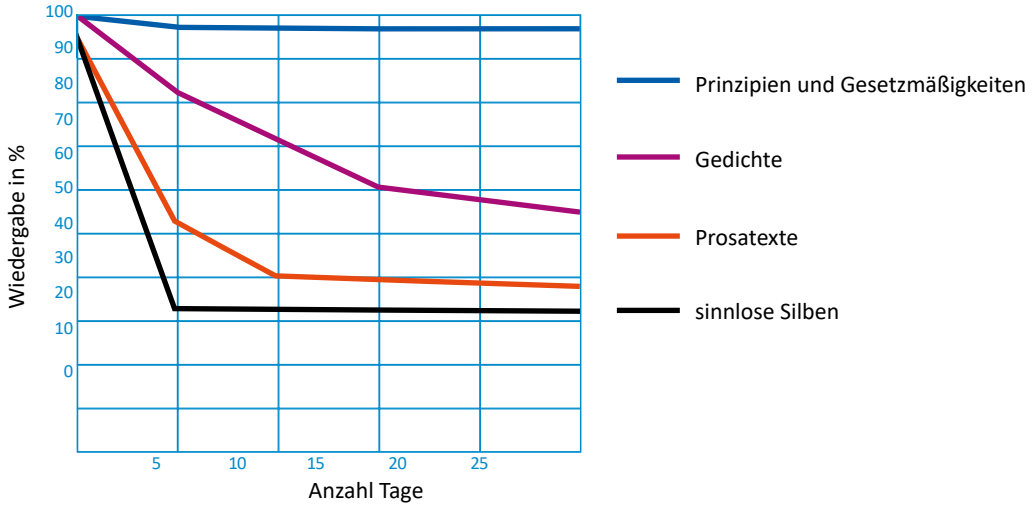


Abbildung 1.5: Vergessenskurve [Mich90] – abhängig von Form und Inhalt des Gelernten

1.4.2 Detailtiefe und Verständnis fehlt

Um ein auf die Kundenanforderungen passendes System zu entwickeln, müssen die gestellten Anforderungen verstanden werden. Einige psychologische Studien legen nahe, dass eine wirkliche Auseinandersetzung mit einer Materie erfordert, dass die kommunizierten Informationen unter anderem niedergeschrieben und visualisiert werden müssen. Diese Auseinandersetzung mit der Materie wird intensiviert, wenn in der Dokumentation ein semiformales Modell (z. B. Diagramme der UML, siehe Kapitel 18 „Anforderungen modellieren“) verwendet wird.

Diese Behauptung lässt sich durch den Aufbau bzw. die Arbeitsweise des menschlichen Gehirns beweisen. Alan D. Baddeley [Baddeley12] beschreibt in seinem Mehrkomponentenmodell des Arbeitsgedächtnisses die Unterteilung in drei verschiedene Module. Die phonologische Schleife ist für die Verarbeitung von sprachbezogenen Informationen verantwortlich und der räumlich-visuelle Notizblock verarbeitet alle visuellen Informationen (z. B. was geschrieben oder visualisiert wurde). Die Koordination beider Informationsspeicher übernimmt die zentrale Exekutive.

Im Jahr 2000 ergänzte Baddeley sein ursprüngliches Modell um den episodischen Speicher. Ursache dafür war die Erkenntnis, dass das Arbeitsgedächtnis mehr Informationen aufnehmen und verarbeiten kann, wenn zwischen sprachlichen und visuellen Inhalten Zusammenhänge bestehen. In diesem Fall kann der episodische Speicher diese Informationen zu komplexen Episoden verknüpfen.

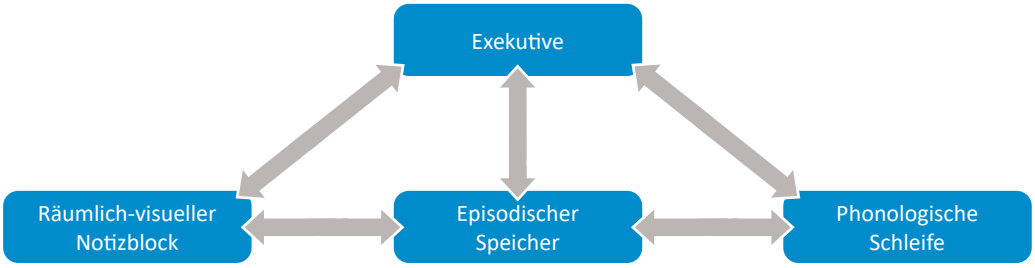


Abbildung 1.6: Arbeitsweise des menschlichen Gehirns

Ein anderer Ansatz von Fergus Craik und Robert S. Lockhart [Spitzer12] aus den 1970er-Jahren legt den Fokus bei der Verarbeitung von Informationen mehr auf die Funktionsweise des Gehirns und das Versenden von Impulsen über Synapsen zwischen den einzelnen Neuronen. Prinzipiell existieren auch bei dieser Sichtweise verschiedene Zentren, z. B. für das Sehen, Hören, Planen, Tasten, Sprechen etc. Die Anzahl der Neuronen und Synapsen, die an der Verarbeitung beteiligt sind, hängt von der Verarbeitungstiefe der Informationen ab, also davon, was mit den Inhalten in den einzelnen Zentren gemacht wird.

Die vorgestellten Theorien belegen, dass bei der Bearbeitung von Problemen deutlich effektiver eine Lösung gefunden werden kann, wenn der auditive und der visuelle Kanal für die Informationsverarbeitung miteinander kombiniert werden. Die intensivste Verarbeitung findet bei der Person statt, die selbst schreibt oder visualisiert. Dabei spielt

es keine Rolle, ob eine Person einen Sachverhalt nur für sich selbst zum besseren Verständnis visuell aufbereitet oder ob mehrere Personen anhand eines Entwurfs über ein Problem diskutieren. Die Ursache dafür ist eine stärkere Ausprägung der Verbindungen zwischen den Synapsen im Gehirn.

1.4.3 Verlust des Gesamtüberblicks

Das menschliche Gehirn kann nur eine begrenzte Anzahl an Informationen gleichzeitig verwalten bzw. verarbeiten. In der Systementwicklung wird diese Kapazität überschritten. Beispielsweise ist die Betrachtung eines Systems als Ganzes aufgrund der Komplexität und des Umfangs in der Regel nicht möglich und eine Zerlegung in einzelne Bestandteile erforderlich. Um den Gesamtüberblick trotz der Zerlegung weiterhin zu bewahren, ist ein gewisses Maß an Dokumentation notwendig. George A. Miller [Miller55] belegte durch verschiedene Experimente, dass das menschliche Kurzzeitgedächtnis in seiner Kapazität stark begrenzt ist und gleichzeitig nur etwa sieben Informationen behalten kann. Diese Begrenzung lässt sich selbst durch Training nicht wesentlich erhöhen. Während seiner Experimente vermittelte Miller den Probanden verschiedene Informationen und ließ sich diese im Anschluss daran wiedergeben. Bei der Wiedergabe wurde ermittelt, wie viele Fehler im Vergleich zu den Ausgangsinformationen enthalten waren. Miller stellte fest, dass bereits ab fünf Informationen (bei Wörtern) die Fehlerrate stark anstieg. Reduzieren lässt sich dieser Mangel nur durch eine hinreichende Dokumentation der Sachverhalte.

1.4.4 Missverständnisse entstehen und bleiben

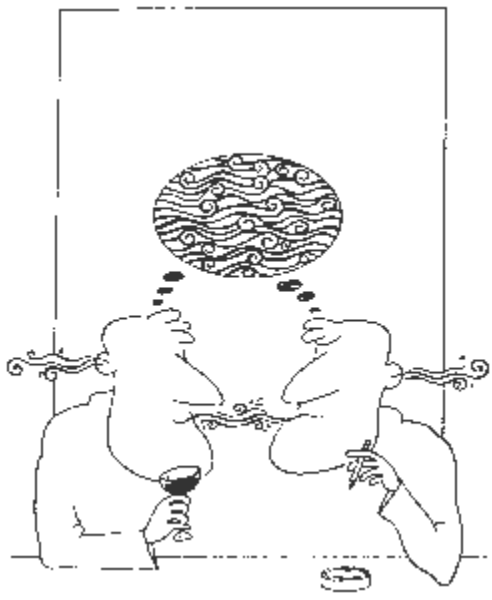
An der Entwicklung eines Systems sind viele verschiedene Personen zur Bearbeitung von unterschiedlichen Aufgaben beteiligt. Aufgrund unterschiedlicher Kenntnisse kann es sehr schnell zu Missverständnissen und Meinungsverschiedenheiten zwischen den Beteiligten kommen. Die Abstimmung der Meinungen und das Festlegen eines gemeinsamen Kenntnisstands sind für den Erfolg der Entwicklung zwingend erforderlich. Die Bildung eines Konsens bzw. die Lösung eines Konflikts zwischen zwei oder mehr Beteiligten ist allerdings nur effektiv möglich, wenn die Parteien ihre Meinung bzw. ihr Wissen explizieren (visualisieren).

Erfolgt die Explikation der jeweiligen Standpunkte nur in verbaler Form, beispielsweise in Diskussionen oder mündlichen Absprachen, dann verändern sich die Modelle in den Köpfen der Beteiligten fortlaufend während der Diskussion, ohne wirklich zu konvergieren. Mögliche Folgen können sein, dass die Diskussion bzw. Verhandlung nicht terminiert oder, wenn sie terminiert, die Modelle der Beteiligten doch nicht übereinstimmen (eben nur in den Vorstellungen). Der Konflikt bleibt weiterhin bestehen.

Genau aus diesem Grund setzen viele Konsolidierungstechniken eine Dokumentation der einzelnen Standpunkte voraus. Sinnvolle Ergebnisse werden in Diskussionen nur erzielt, wenn alle Parteien ihre Argumente zum gleichen Modell äußern und nicht nur zu dem in ihren Vorstellungen.

Ein weiteres Beispiel für die sinnvolle Umsetzung dieser Behauptung in der Praxis ist das Prototyping. Dabei liefert das Entwicklungsteam eine frühe Version eines Systems aus, um festzustellen, ob die Anforderungen der Kunden richtig verstanden wurden. Da das Prototyping für Verhandlungen zwischen auftraggebenden und auftragnehmenden Unternehmen zeitaufwendig und kostenintensiv sein kann, kann die schriftliche Dokumentation der Vorstellungen beider Parteien als Ersatz dafür verwendet werden.

1.4.5 Abweichende Informationen verteilen sich



Zur erfolgreichen Entwicklung eines Systems müssen die vorliegenden Informationen in einem Entwicklungsvorhaben an verschiedene Personen verteilt werden. Allerdings kann nur Wissen, welches persistiert ist (also z. B. niedergeschrieben, gemalt oder in Form eines Videos existiert), ohne Abweichungen vonseiten der Sendenden an eine größere Anzahl von EmpfängerInnen verteilt werden (sofern diese nicht alle zum Übertragungszeitpunkt in einem Raum sitzen oder per Telefon- oder Videokonferenz verbunden sind oder die Übermittlung per Video dokumentiert wird). Bei mündlicher Verteilung ist zudem ausschließlich eine synchrone Übertragung möglich. Das heißt, beide beteiligte Parteien müssen interagieren. Bei der schriftlichen Weitergabe von Informationen durch Dokumente sind diese unerwünschten Veränderungen der Inhalte durch Mimik, Gestik, verwendete Sprache und Tonfall nicht gegeben. Ein Dokument ändert sich mit der Vervielfältigung nicht.

Verstehen Sie die Argumente jetzt bitte nicht als Hinweis, dass Sie alles möglichst umfangreich dokumentieren sollten. Wägen Sie den Nutzen und den Aufwand einer jeden Dokumentation ab.

1.5 Typische Probleme im Requirements-Engineering

Was kann der Requirements-Engineer von der Philosophie lernen? Wesentlich mehr, als man angesichts der Verschiedenheit der Disziplinen vielleicht denken möchte. Folgen Sie im Artikel „Erkenntnistheorie, (Wirtschafts-)Informatik und Requirements-Engineering“ unter www.sophist.de/re7/kapitel1 Prof. Dr. Holl auf einen spannenden interdisziplinären Exkurs über Modelle der Realität in der Philosophie und der Informatik und den verbesserten Umgang mit ihren gemeinsamen Schwächen. Das, was Sie als Requirements-Engineer erstellen, ist nämlich ein Modell der Realität, das den Lösungsmodellraum aufspannt, in dem sich später die ArchitektInnen austoben können.

Um Sie ausreichend zum Lesen der folgenden Kapitel zu motivieren, die Ihnen ein professionelles Requirements-Engineering näherbringen, mache ich Sie noch mit den typischen Problemen eines mangelhaften Requirements-Engineerings bekannt. Die im Requirements-Engineering entstehenden Probleme lassen sich systematisieren und auf die in Abbildung 1.7 gelisteten Hauptprobleme herunterbrechen.

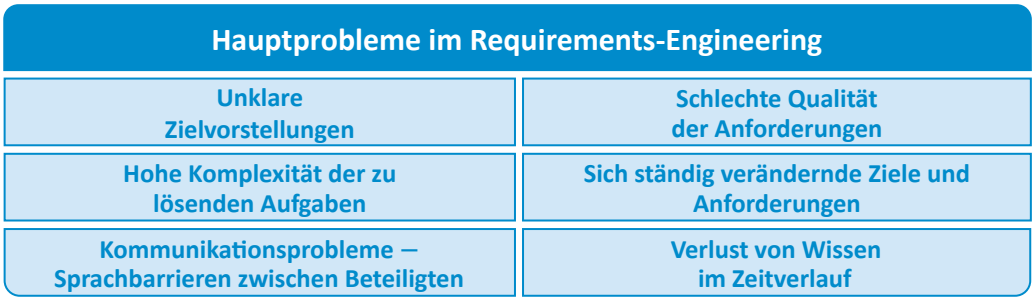


Abbildung 1.7: Darunter leidet das Requirements-Engineering am meisten

Die Antwort, wie Sie diese Probleme in den Griff bekommen, liefert Ihnen dieses Buch in den folgenden Kapiteln. Wir wünschen Ihnen jetzt schon viel Spaß beim Lesen und Lösen Ihrer Probleme.

Requirements are a Socio-Technical Discipline

We are often asked „What makes a good requirements analyst?“ The short answer is willingness to listen, but it is worth looking a little further at the nature of the requirements activity to find a better answer to the question.

Requirements must be thought of as an activity that straddles the boundary between the sociological side of system development, and the technological side. On one hand we have people, with all their vagaries and fallibilities. On the other we have technology that demands a precise specification if the developers are to bring the best possible solution to the client. There are several significant aspects to the sociological side of the activity. Firstly, the requirements analyst must identify and involve all the appropriate stakeholders to discover all requirements. Also consider that some stakeholders are too busy to pay proper attention, some don't know enough to supply the right requirements, and some think they know but don't.

What about the technological side of the fence? The skilled business analyst must know enough about the technology to know what is possible. People don't ask for things unless they know the things exist, or they have a good probability of being able to exist. So it falls to the business analyst to invent part of the system. If the analysts simply listened to their customers, then

not only would each generation of system look pretty much like the previous ones, but few genuine advances would be made.

Why is it important to see requirements analysis as a socio-technical discipline? Because software has become a commodity. There are too many people producing it, and too many people competing for your clients' software business. It is simply too risky to leave the requirements – the most important part of the development cycle – to chance.

James and Suzanne Robertson are the founders of the Volere requirements process, template and checklists. This acclaimed requirements technique is used by tens of thousands of organizations worldwide. Their careers have taken them to every continent and along the way they have collected an impressive portfolio of projects and industries. They can be reached through the Atlantic Systems Guild, a London, New York and Aachen consultancy and think tank.

www.systemsguild.com

Books:[DeMarco08], [Robertson12], [Robertson04], [Robertson98]