

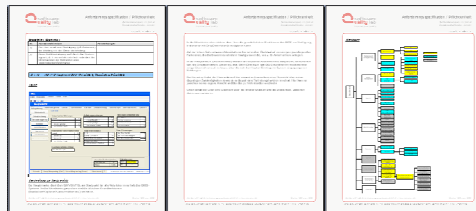
In dieser Ausgabe:

- ⇒ Detailliertheit der Requirements-Spezifikation
- ⇒ Requirements-Engineering Grundlagen
- ⇒ Requirements-Engineering im agilen Umfeld
- ⇒ Tipps & Tricks für gute Requirements

Detailliertheit der Requirements-Spezifikation

Sehr oft wird die Frage gestellt: **Wie detailliert soll der Auftraggeber seine Anforderungen spezifizieren?**

Es gibt eine einfache Antwort auf diese Frage:



Alles was dem Kunden bzw. Auftraggeber wichtig ist, muss als seine Anforderung spezifiziert werden!

Um dies etwas zu verdeutlichen: Wenn einem (nur) wichtig ist, dass man z.B. eine „Kundenverwaltung“ bekommt, dann genügt es wie folgt zu spezifizieren: „Das System muss eine Möglichkeit bereitstellen, die Kunden zu verwalten“. Wenn einem Kunden jedoch wichtig ist, rechts oben das Firmenlogo zu haben, auch lange Familiennamen mit bis zu 150 Zeichen eingeben zu können und vor dem Speichern noch eine Dublettenprüfung durchzuführen, dann müssen all diese Punkte explizit spezifiziert werden, denn sonst hat er als Auftraggeber (rechtlich) keinen Anspruch darauf!

Es hat sich in vielen Projekten gezeigt, dass folgende Detaillierungsebenen für eine gute Qualität der gesamten Abwicklung (Architektur, Entwicklung, Test, Abnahme) hilfreich sind:

1. **Business-Prozesse** und deren Varianten
2. **Use-Cases**, jedoch nicht nur rein textuell, sondern ergänzt durch Aktivitäten-, Interaktions-, Zustands-Diagramme, etc.
3. (Grobes) **Layout/Prototyping der Masken und Reports** (Tools zum Erstellen sind genügend vorhanden)
4. Genaue **Beschreibung aller Schnittstellen** zu Fremdsystemen
5. Ergänzende **funktionale Detailbeschreibung** (in der Art eines Benutzerhandbuchs, das schon vorab erstellt wird) aller Prozessen, Masken, Reports und Schnittstellen
6. **Klar prüfbare (!) nicht-funktionale Anforderungen** nicht vergessen, auch wenn sie oft nicht ganz einfach zu formulieren sind!



... denn sie wissen nicht (bzw. wollen nicht wissen), was sie wollen !

In meiner Rolle als Projekt-Auditor oder Projekt-Qualitätssicherer ist es immer wieder meine Aufgabe, die Qualität von Spezifikationen (z.B. User-Requirements, Lastenheft, Pflichtenheft, ...) sicher zu stellen.

Ein wesentlicher Qualitätsaspekt von Spezifikationen ist die Vollständigkeit bezüglich der Kundenwünsche und -vorstellungen.

Leider gibt es viele Kunden bzw. Auftraggeber, die nicht wissen, was sie (inhaltlich) wollen bzw. sich nicht die Zeit nehmen, darüber ernsthaft nachzudenken.

Teilweise ist dies dadurch begründet, dass der Kunde zu wenig (Fach- bzw. Technologie-)Wissen in dem zu beauftragenden Bereich besitzt. Oft nimmt sich der Kunde aber nicht ausreichend Zeit, um seine Vorstellungen und Wünsche zu spezifizieren. In manchen Fällen ist es auch einfach nur „Bequemlichkeit“.

Vielfach wird dann der Lieferant mit der Spezifikation der Kundenwünsche beauftragt. Doch wie soll dieser wissen, was der Kunde will und braucht?

Im Laufe des Projekts zeigt sich dies dann oft durch Aussagen des Kunden wie z.B. „So hab ich mir das aber nicht vorgestellt!“ oder eine nicht kalkulierte Change-Request-Organie mit Streitereien darüber, wer nun die Kosten dafür übernimmt.

In diesem Sinn kann man bezüglich der Requirements-Ermittlung in vielen Projekten in Abwandlung von bekannten Zitaten feststellen: „Zeige mir wie Dein Projekt beginnt und ich sage Dir, wie es endet!“

Dipl.-Ing. Johannes Bergmann

Staatl. befugter und beedeter Ingenieurkonsultent für Informatik

Der Quality-Knowledgeletter ist ein periodisches Informationsmedium von Software Quality Lab und dessen Partnern mit den Schwerpunkten IT-Qualitätsmanagement, Projekt- und Prozess-Management.

Inhalt: fachliche Beiträge und Schwerpunktthemen, Vorstellung neuer Produkte und Leistungen, neue wissenschaftliche Erkenntnisse und andere Fachbeiträge aus unseren Themenbereichen.

Aktuelle Fach- und Forschungsbeiträge sind willkommen. Einsendungen an info@software-quality-lab.at.

Weitere Infos zu diesem und anderen Themen finden Sie auf <http://www.software-quality-lab.at>.

Requirements-Spezifikation - Ein kurzer Grundlagen-Auszug

Software Quality Lab

Definitionen

Requirement

Ein Requirement ist eine Aussage über eine zu erfüllende Eigenschaft oder zu erbringende Leistung eines Produktes, Systems oder Prozesses. [Wikipedia]

(1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a product or product component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2). [IEEE 610.12-1990]

Pragmatische Definition: „Die Beschreibung, WAS das spezifizierte System leisten soll?“

Hinweis: Im Detail wird noch unterschieden zwischen Kunden-/Anwender-Anforderungen, System-Anforderungen und Architektur/technischen Anforderungen, etc. In diesem Artikel wird auf die Kunden/Systemanforderungen eingegangen und nicht auf die „technischen“ Anforderungen.

Stakeholder = Beteiligter oder Betroffener. Es ist wesentlich für den Projekterfolg, dass für die Anforderungsspezifikation möglichst alle Stakeholder einbezogen werden!

Requirements-Provider = der/die Stakeholder, die eine Anforderung definiert haben bzw. das Entstehen dieser Anforderung bewirkt oder verantwortet haben.

Motivation

Die Kosten für Fehlersuche und Fehlerbehebung in Entwicklungsprojekten betragen oftmals bis zu 30-50% der Gesamtkosten.

Die nachfolgende Grafik zeigt, dass dabei mehr als 50 % der Fehler in der Spezifikationsphase entstehen.

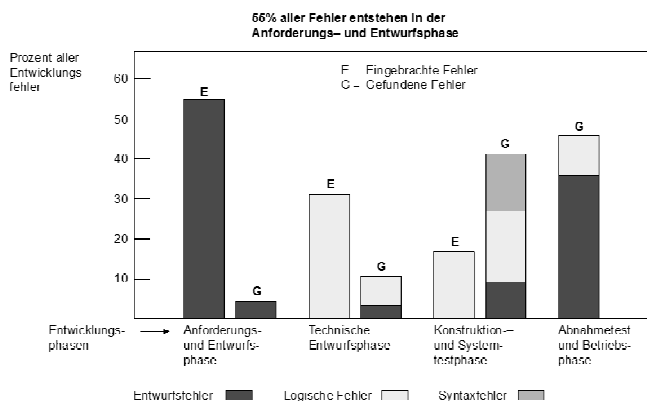


Abb. 1 - Fehler-Entstehung & -Entdeckung [Quelle: Balzert]

Der überwiegende Anteil dieser Fehler wird jedoch leider erst zur Abnahme bzw. im Betrieb gefunden!

⇒ Eine Fehlervermeidung bereits in der Spezifikations-Phase ist am effektivsten!

Requirements-Prozesse

Es gibt zwei Bereiche, die für die Requirements-Spezifikation von zentraler Bedeutung sind:

1. Requirements-Engineering: In diesen (Teil-)Prozessen entsteht die Requirements-Spezifikation.
2. Requirements-Management: In diesen (Teil-)Prozessen wird die Requirements-Spezifikation im Laufe des Projekts weiterentwickelt bzw. kontrolliert, nachvollziehbar verwaltet und verändert.

Die Teilprozesse dieser beiden Prozess-Bereiche sind nachfolgend schematisch im Überblick dargestellt:

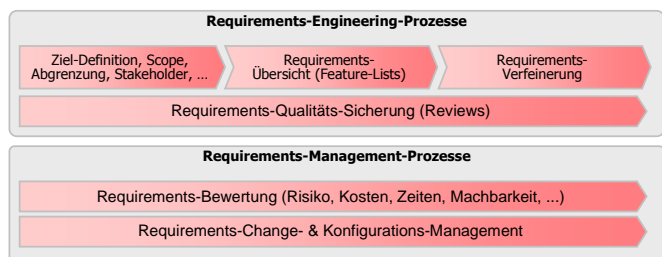


Abb. 2 - Requirements-Engineering & -Management Prozesse [Quelle: SWQL]

Festgehalten werden muss, dass Requirements-Engineering jedenfalls nicht sequentiell, sondern ein **inkrementeller, iterativer Prozess** ist (auch wenn dies aus der obigen vereinfachten Prozessdarstellung nicht direkt hervorgeht).

Die Anforderungen sind im Normalfall nicht in ausreichender Detaillierungstiefe beim Kunden vorhanden und entwickeln sich Schritt für Schritt bzw. Phase für Phase im Projekt.

Wichtig und von entscheidender Bedeutung ist dabei eine strukturierte und methodisch unterstützte Vorgehensweise.

Diverse Hilfsmittel, um in einem nachvollziehbaren Prozess zu guten Requirements zu kommen sind sinnvoll und erwünscht (z.B. Prototyping, Simulation, agile bzw. interaktive Spezifikationserstellung, etc.).

Requirements-Darstellung

Der Einsatz geeigneter Anforderungs-Gewinnungs- und Darstellungstechniken ist wichtig für eine kundenorientierte Entwicklung der Anforderungsspezifikation.

Hier muss besonders in den ersten Iterationen, in denen möglicherweise technisch nicht so versierte Personen des Kunden eingebunden sind, darauf geachtet werden, dass die Darstellung der Anforderungen kundengerecht und nicht zu technisch erfolgt und dass aus Anwendersicht und zusammen mit den Anwendern spezifiziert wird.

Darstellungstechniken wie Prozess-Diagramme, Tabellen, Bildschirm-Darstellungen, Baumstrukturen, Netzwerke, ... welche eine rein textuelle Beschreibung ergänzen und auflockern und einen schnellen Überblick schaffen, haben

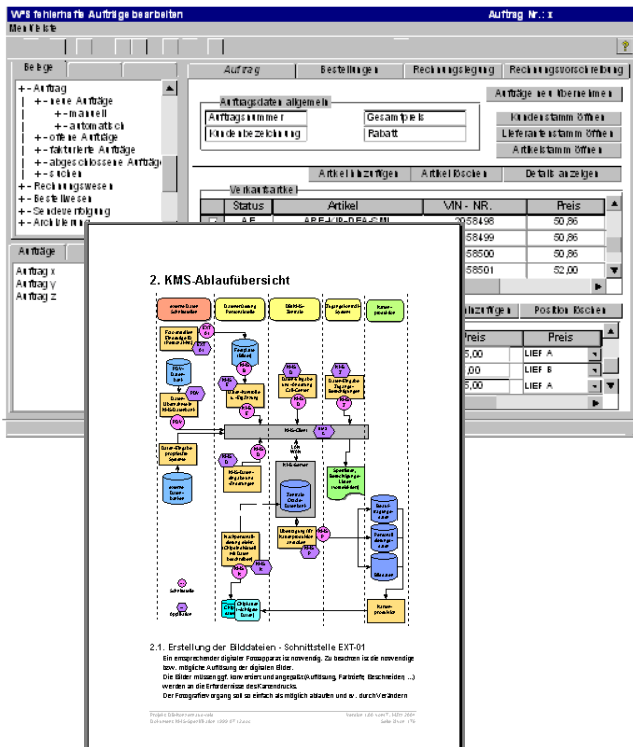


Abb.3 - Requirements- & Prozess-Darstellung—Beispiele
[Quelle: Bergsmann]

sich bei Software Quality Lab in den Projekten bestens bewährt und es sollte daher nicht darauf verzichtet werden.

Technische Problemstellungen und Details, die nicht Bestandteil der Aufgabenstellung und nicht notwendig für die verbindliche Anforderungsvisualisierung sind, sollten nicht (mit oder von dem Kunden) in der Kunden-/System-Anforderungs-Spezifikation modelliert und spezifiziert werden (Ausnahme: Wenn es für das Verständnis oder für den Zusammenhang wichtig ist).

Spezifikation ist immer auch Konsensbildung, da die Sichten von unterschiedlichen Stakeholdern und Requirements-Providern und deren unterschiedliche Anforderungen und Priorisierungen zu Konflikten zwischen den Anforderungen führen können, die im Rahmen der Spezifikation aufgelöst werden müssen.

Arten von Requirements (Grobklassifikation)

Funktionale Anforderungen: legen fest, was das System tun soll und wie es sich darstellt.

Typische Requirements-Arten in diesem Bereich sind Eingabemasken, Ausgaben/Berichte und Schnittstellen zu anderen Systemen.

Beispiele: „Das System soll die Stamm-Daten von Patienten verwalten.“ oder „Das System soll einen Report über die durchgeführten Untersuchungen ausgeben.“

Nichtfunktionale Anforderungen (Qualitätsanf.): legen fest, welche Eigenschaften und Qualität das System haben soll. Typische Requirements-Arten in diesem Bereich sind Usability, Performance, Fehlertoleranz, Kompatibilität, Wartbarkeit, ...

Sie beeinflussen meist auch stark die Gestaltung und Architektur (äußere und innere Qualität) des Systems.

Beispiele: „Das System soll dem Anwender innerhalb von 2 Sekunden antworten.“ oder „Die Daten müssen vor unberechtigten Zugriffen geschützt sein.“

Zusätzlich werden oft auch **Rahmenbedingungen** als eigene Art beschrieben, wobei dies keine durch das Team modellierbaren Anforderungen sind, da sie ja nicht explizit gefordert werden, sondern von außen vorgegeben sind.

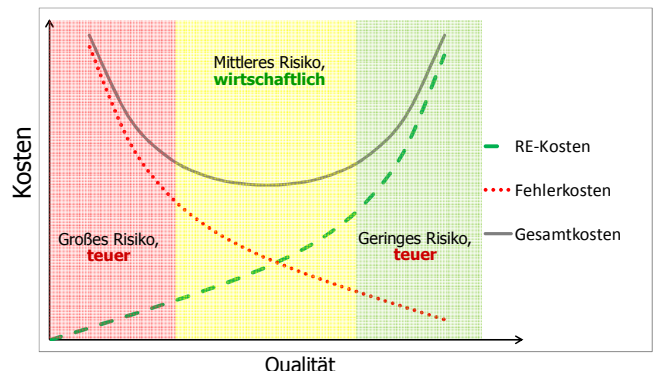
Beispiele: Obergrenze für die Kosten, einzuhaltende Termine, eine bestimmte zu verwendende Technologie.

Hinweis: Die Beispiele sind aus Platzgründen nur (zu) grobe Anforderungen, die in Projekten noch entsprechend weiter verfeinert werden müssen!

Wie detailliert soll spezifiziert werden?

Eine zu grobe Spezifikation birgt ein hohes Fehlerkostenrisiko, da damit gerechnet werden muss, dass viele Nacharbeiten und Korrekturen nötig sein werden, bis der im Kopf des Auftraggebers vorgestellte Zustand vom Auftragnehmer dann tatsächlich umgesetzt wird.

Eine zu detaillierte Spezifikation birgt die Gefahr, dass das Projekt nicht über die Spezifikationsphase hinaus kommt, da sich schon in der Spezifikation immer wieder Details ändern und dadurch die Spezifikationsphase zu



keinem Ende kommt, was weiters zu unverhältnismäßig hohen Spezifikationskosten führt.

Am sinnvollsten ist - wie in vielen Lebensbereichen - ein gesundes Mittelmaß.

Die Spezifikation sollte so detailliert sein, dass ein erfahrender Entwickler guten Gewissens eine Aufwandsschätzung für die Realisierung mit einer Ungenauigkeitswahrscheinlichkeit von max. 10-20% abgeben kann und auch der erfahrene Tester daraus sinnvolle Testfälle für den Test spezifizieren kann.

Aus der Erfahrung hat sich gezeigt, dass es sinnvoll und noch wirtschaftlich ist, einen Aufwand von ca. 20-25% des geplanten Gesamt-Projekt-/Produkt-Budgets in die Spezifikation zu investieren.

Wie weit muss der IST-Zustand ermittelt werden?

Eine Beschäftigung mit dem IST-Zustand sollte dann erfolgen, wenn dies notwendig ist, z.B. zum Verstehen der Arbeit und der Bedürfnisse der Benutzer, zur Ermittlung von

Stärken und Schwächen eines bestehenden Alt-Systems (falls diese nicht schon bekannt sind) oder zur Übernahme von Teilen eines IST-Systems.

Eine zu intensive Beschäftigung mit dem IST-Zustand birgt die Gefahr der Orientierung am Bestehenden und verursacht möglicherweise eine gewisse Betriebsblindheit. Der bestehende Zustand soll ja durch ein neues System typischerweise verbessert werden und nicht eingefroren!

Wenn der IST-Zustand ausreichend gut analysiert wurde, kann dieser gegebenenfalls auch als Basis für eine Anforderungsspezifikation verwendet werden.

Wenn der IST-Zustand zwar verwendet werden soll, jedoch noch nicht ausreichend erhoben wurde, ist darauf zu achten, dass nicht das oft übliche „Killer-Requirement“ verwendet wird: **„Das neue System soll alles das können, was das alte System auch kann, sowie zusätzlich noch ...“**

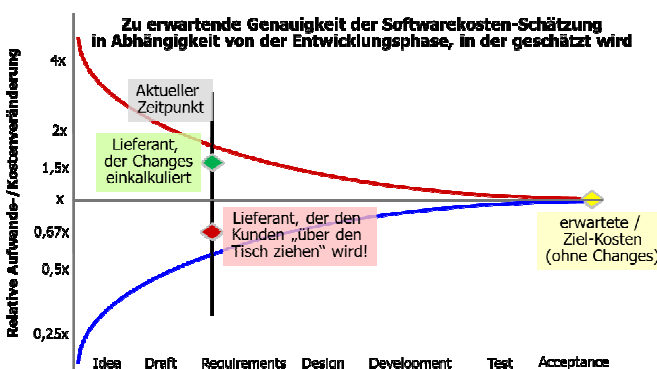
Denn wie soll der arme Lieferant ohne weitere Information wissen, was das alte System alles kann/konnte?

Warum gewinnt bei Festpreis-Ausschreibungen mit schlechten Spezifikationen immer der schlechte Lieferant?

Bei Ausschreibungen wird meist vom Auftraggeber ein Festpreisangebot erwartet, das dann gemeinsam mit der ausgeschriebenen Spezifikation die Vertragsgrundlage für die Abwicklung bildet.

Da die Spezifikation des Auftraggebers jedoch meist unklar formuliert ist, gewinnt bei Ausschreibungen immer der „schlechte“ Lieferant.

Warum dies so ist, erklärt nachfolgende Grafik (Unschärfe-Trichter von Böhms):



Es ist Faktum, dass der Preis bei einer ungenauen Spezifikation auch mit einer gewissen Unschärfe kalkuliert werden muss.

Im Beispiel sei der erwartete Zielpreis 200.000 EUR.

Der „gute“ Lieferant wird daher in diesem Fall auf den zuerst kalkulierten bzw. erwarteten Zielpreis einen Sicherheitszuschlag dazugeben, da er ja damit rechnen muss, dass vom Auftraggeber entsprechend viele Change-Requests geltend gemacht werden, von denen der Auftraggeber behaupten wird, dass dies aus seiner Sicht natürlich im Preis enthalten sein muss.

Der angebotene Preis dieses Lieferanten wird daher z.B. mit Sicherheitszuschlag bei 290 T. Euro liegen.

Der „schlechte“ Lieferant, der den Kunden bewusst „über den Tisch ziehen“ möchte, wird überlegen, dass er mit einem Sicherheitszuschlag den Auftrag auf keinen Fall bekommen wird, da er dann für den Auftraggeber zu teuer ist.

Er wird daher in diesem Fall jedenfalls einen niedrigeren Preis als den erwarteten Zielpreis angeben und damit rechnen, dass vom Auftraggeber entsprechend viele Change-Requests geltend gemacht werden, die er dann gegen ein entsprechend hohes Zusatzentgelt verrechnen kann und sich damit dann den gewährten Preisnachlass vom Auftraggeber im Laufe des Projekts wieder zurückholt.

Der angebotene Preis dieses Lieferanten wird daher z.B. bei 160 T. Euro liegen.

Welcher Lieferant gewinnt nun die Ausschreibung?

Natürlich der mit dem geringsten Preis, da ja beide Lieferanten den Preis auf Basis derselben (unklaren) Spezifikation abgegeben haben und damit scheinbar dieselbe Auftragsgrundlage besteht.

In der Praxis wird der Auftraggeber in diesem Fall bald merken, dass das Projekt preislich völlig aus dem Ruder läuft und im Endeffekt genau so viel oder mehr gekostet hat, wie beim guten Lieferanten.

Nur dass der Auftraggeber nun als „Strafe“ für diese Fehlentscheidung zum erhöhten Preis auch noch Streitereien mit dem Lieferanten ertragen muss, unnötigem Projektstress wegen meist dann auch zeitlicher Fehleinschätzungen hat, sowie eventuell nachträgliche Budget-Erhöhungen gegenüber seinen Geldgebern und Entscheidungsträgern argumentieren muss und im schlimmsten Fall kann es sogar zu einem Projektabbruch kommen.

Argumente für die Spezifikation von Requirements

Es stellt sich immer wieder die Frage, warum denn eine Spezifikation überhaupt nötig ist und was dies denn wirtschaftlich bringt? Dafür gibt es unter anderem folgende Argumente:

Gegenüber den **Entscheidungsträgern** kann gesagt werden, dass dadurch **Kosten gesenkt** werden können:

- geringere Herstellungskosten (durch Senken der Fehlerkosten!)
- Vermeidung von unkontrolliertem „Requirements-Wildwuchs“ (Eindämmen von Change-Requests)
- geringere Wartungs- und Pflegekosten

Für den **Vertrieb** ist wichtig, dass dadurch **mehr verkauft** werden kann:

- Weniger Probleme durch nachträgliche Änderungen
- zufriedener Kunden

Die **Projektmanager** profitieren davon, dass **die Projekte strukturierter sind und besser beherrscht** werden können:

- Risiken besser erkennen und verkleinern
- Zuverlässigere Prognosen für Termine
- Realistischere Kosten-Schätzungen

Requirements-Engineering im agilen Umfeld

Johannes Bergsmann

In den agilen Methoden, wie z.B. Scrum, wird das Thema Requirements-Spezifikation oft zu wenig angesprochen. In anderen Methoden wird dies nur unzureichend angesprochen (z.B. Erstellung von User-Stories). Damit sind viele Mitarbeiter in agilen Projekten und Teams verunsichert und spezifizieren oft gar nicht oder unzureichend. Dieser Artikel stellt einige Aspekte des Requirements-Engineerings (RE) im agilen Umfeld dar.

Braucht es im agilen Umfeld überhaupt ein Requirements-Engineering?

Dies kann ganz klar mit „Ja“ beantwortet werden.

Auch wenn es bei manchen Verfechtern der agilen Methoden auch verpönt ist, so ist z.B. das Erstellen von User-Stories, das Erstellen von User-Interface-Prototypen (Mock-Up), das Klären von unklaren Anforderungen mit dem Auftraggeber (Product-Owner) und die Feinplanung von Features z.B. im Rahmen des Sprint-Plannings ganz klar ein Teil von Requirements-Engineering.

Was soll im agilen Umfeld spezifiziert werden?

Da in agilen Projekten das Kunden-Feedback meist intensiv und in kurzen Abständen erfolgt, ist hier innerhalb einer Iteration schon vom Grundprinzip her keine Spezifikation bis ins letzte Detail notwendig.

Allerdings werden wesentliche Aspekte oft unterschätzt oder vernachlässigt:

- Top-Down-Überblick:** Oft werden in agilen Projekten auch „agile“ Tools zur Spezifikation verwendet (z.B. werden häufig Tracking-Systeme oder kleine Notiz-Zettel dazu verwendet). Genauso schaut dann aber auch die Spezifikation aus - nämlich eine mehr oder weniger geordnete Liste von unzähligen kleinen Requirements (Features). In vielen Projekten wissen die Beteiligten dann nach kurzer Zeit nicht mehr, was eigentlich schon alles umgesetzt wurde und verlieren den Überblick. Es entsteht ein Wildwuchs an Funktionen und dadurch viele unnötige Iterationen, in denen das System wieder einem „Refactoring“ unterzogen werden muss. Hier fehlt ganz klar die übergreifende Top-Down-Sichtweise auf das System, die leider bei einer Fokussierung auf die User-Stories oft verloren geht. Daher sollte darüberhinaus noch eine **Top-Down-Spezifikation** existieren, in der die **Projektziele**, die übergeordneten **Business-Prozesse** des Systems, der **Schnittstellen-Kontext**, etc. im Kontext dargestellt werden (am besten effizient durch ein passendes Requirements-Tool unterstützt).

- Kenntnis über die Gesamtfunktionalität des Systems:** Aus den vorher erwähnten Gründen passiert es in vielen agilen Entwicklungen (vor allem längerfristige Produktentwicklung), dass sowohl der Entwickler, als auch der Auftraggeber (Product-Owner/Manager) bald nicht mehr wissen, was denn das System insgesamt kann. Das ist dann in vielen Bereichen sehr problematisch: Es gibt ev. keine Gesamtdokumentation der Funktionalität für den Kunden und der Entwickler implementiert ev. verschiedene Features mehrfach, obwohl diese schon vorhanden sind. Bei umfassenderen Neuentwicklungen des Produkts weiß man nicht mehr, was denn das bisherige Produkt alles konnte, etc. und sich das im Anlassfall aus dem Source-Code mühsam zusammen zu suchen, ist dann oft auch zu aufwändig. Es ist daher zu empfehlen, auch in agilen Projekten eine konsistente Gesamt-Dokumentation des Systems zu führen (sei es als Benutzer-Dokumentation oder auch eine interne Systemdo-

kumentation). Dabei muss jedoch beachtet werden, dass diese Dokumentation erst dann erstellt bzw. aktualisiert wird, wenn sich die entsprechenden Systemteile bzw. Features schon etwas stabilisiert werden (z.B. im Nachhinein in der 2.-4. Folgeiteration), da sonst der Änderungsaufwand der Dokumentation zu groß wird.

Wird mit agilem Vorgehen bei unklarer Spezifikation das Projektziel günstiger und schneller erreicht?

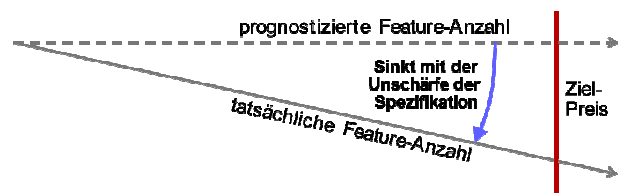
Die Behauptung steht bei agilem Vorgehen im Raum, dass durch die enge Kundenbindung das Projektziel trotz anfänglich unklarer Spezifikation effizienter und schneller erreicht werden kann.

Faktum ist, dass wir in einer Welt der begrenzten Ressourcen (Personen, Zeit, Geld) leben und dies auch für die agilen Vorgehensweisen gilt. Wenn also eine Spezifikation anfangs unklar ist, dann führt dies zwangsläufig dazu, dass auch mehr Aufwand (Iterationen) notwendig ist, um das eigentliche Ziel des Auftraggebers zu erreichen.

Es wird nun oft argumentiert, dass in agilen Methoden das geplante Budget des Auftraggebers auf jeden Fall eingehalten werden kann, da der Auftraggeber ja immer am Schluss der letzten Iteration ein lauffähiges Produkt bekommt.

Was oft nicht bedacht wird oder dem Auftraggeber verschwiegen wird ist, dass er **bei Einhaltung eines definierten Zielpreises** zwar ein lauffähiges Produkt bekommt, dass jedoch **dann möglicherweise noch nicht alle von ihm gewünschten Features zu diesem Zeitpunkt realisiert** wurden.

Wenn der Auftraggeber nun auf seinen Feature-Wünschen besteht, müssen zusätzliche Iterationen eingeschoben werden, was das Projekt nun auch wieder teurer macht.



Diese Abweichung wird umso größer, je ungenauer die Spezifikation am Beginn der übergreifenden Planung ist.

Es gilt daher abzuwägen inwieweit man nicht auch bei agilen Methoden im Vorfeld in eine Spezifikationsphase investieren sollte, damit die Gesamt-Projekt-Abweichung in einem „erträglichen“ Rahmen bleibt. Dabei sollte die Spezifikation jedoch nicht so weit gehen, dass die Vorteile der agilen Entwicklung zunichte gemacht werden. Dies ist mitunter eine schmale Gratwanderung.

Im Endeffekt wird jedoch der Preis und die Zeitdauer bei einem vom Auftraggeber gewünschten Feature-Umfang meist nicht durch die gewählte Methode (agil oder klassisch) bestimmt, sondern vorwiegend durch die Qualität der vorangehenden Spezifikation und das Engagement des Auftraggebers im Projekt.

Tipps & Tricks für gute Requirements

Software Quality Lab

Requirements gut zu formulieren ist keine leichte Aufgabe und bedarf in vielen Fällen auch intensiver Übung mit einem erfahrenen Requirements-Engineer oder Analytiker. Nachfolgend sind einige Vorgehensweisen, Eigenschaften und Richtlinien angegeben, die bei der Formulierung von guten Requirements beachtet werden sollten.

Allgemeine Hinweise

Erarbeiten Sie die Anforderungen in mehreren Stufen / Iterationen, in denen Sie die einzelnen Anforderungen immer weiter verfeinern (wenn Sie kein Requirements-Management-Tool verwenden, sind hier oft Mindmaps oder Baumstrukturen für die Gliederung hilfreich), bis (dem Requirements-Provider) klar ist, wie jedes einzelne Requirement in der Abnahme geprüft werden soll.

Es hat sich bewährt, zuerst die Projekt-Ziele und die Prozesse sowie die dazugehörigen Detailabläufe zu überlegen und davon ausgehend die Anforderungen weiter zu spezifizieren.

Der Detaillierungsgrad wird durch den Requirements-Provider gesteuert. Alles was dem Requirements-Provider wichtig ist und in bestimmter Art und Weise realisiert werden soll, muss auch als Anforderung von ihm spezifiziert werden.

Dies kann sehr grob sein wie z.B. „Das System soll Kunden verwalten können.“ wenn es dem Requirements-Provider unwichtig ist, was bei der Verwaltung im Detail passiert (Anm. das ist meist jedoch unwahrscheinlich) oder aber auch sehr detaillierte Systemeigenschaften wie z.B. „Das Feld Vorname muss mindestens 30 Zeichen lang sein und rechtsbündig und in roter fetter Schrift dargestellt werden.“, wenn er genau auf das bei der Abnahme Wert legt.

Jene Anforderungen, die spezifiziert werden, sollten immer auch von anderen Beteiligten/Betroffenen noch einmal einem Review unterzogen werden.

Es sollten möglichst alle Stakeholder bzw. deren Vertreter in die Spezifikationserstellung einbezogen werden (zumindest als Reviewer).

Jedes einzelne Requirement soll ...

- ⇒ mindestens einem Requirements-Provider zugeordnet sein. Wenn es keinen gibt, warum sollte diese Anforderung dann realisiert werden?
- ⇒ klar und verständlich formuliert sein (ggf. Prototyping-Techniken verwenden).
- ⇒ prüfbar sein. Auf welcher Basis soll sonst eine Abnahme des Systems erfolgen?
- ⇒ verstanden werden - sowohl vom Auftraggeber als auch vom Auftragnehmer und allen betroffenen Stakeholdern.
- ⇒ identifizierbar sein (wenn möglich eindeutig nummerieren, damit die Anforderung in anderen Dokumenten (z.B. Umsetzungsdokumente, Testspezifikation, ...) referenziert werden kann).
- ⇒ positiv formuliert sein (negative Formulierungen wie „nicht“, „nie“, ... können meist nicht geprüft werden).

Die Requirements-Spezifikation insgesamt soll ...

- ⇒ in sich konsistent sein (keine Widersprüche zwischen einzelnen Anforderungen bzw. Widersprüche sollten vor Freigabe klar aufgelöst werden)
- ⇒ vollständig sein. Es sollte möglichst keine „impliziten“ Anforderungen mehr geben (welche die einzelne Stakeholder im Kopf haben und die noch nicht erfasst wurden).
- ⇒ nicht nach dem ersten Wurf freigegeben werden, sondern in zumindest 3-4 Iterationen verfeinert werden:
 - Ziele
 - Business-Prozesse
 - Use-Cases und Detail-Workflows
 - Features / Requirements-Übersicht
 - Detailformulierungen für die Features und User-Interface-Prototyping

Es gibt noch verschiedenste weitere Kriterien und Richtlinien, die jedoch hier aus Platzgründen nicht angeführt werden.

Zusammenfassend kann Requirements-Engineering als ein für den Erfolg in allen Arten von Projekt-Vorgehensweisen unbedingt notwendiger Prozess gesehen werden, der viel Erfahrung und Feingespür für das richtige Maß benötigt und jedenfalls auch ein hohes Engagement des Auftraggebers erfordert.

Leistungen

Im Rahmen des **Requirements-Engineering & Requirements-Management** unterstützen die Experten von Software Quality Lab durch folgende Leistungen:

- ⇒ **Erstellung** von Spezifikationen
- ⇒ Definition von **Anforderungen** und Entscheidungsgrundlagen
- ⇒ Requirements-Engineering-**Seminare**
- ⇒ **Prüfen / Review** von Spezifikationen
- ⇒ Unterstützung bei der Erstellung von **Vorlagen und Checklisten** für Spezifikationen, ...
- ⇒ **Ausbildung** von Requirements-Managern
- ⇒ Begleitendes Projekt-**Coaching** nach Bedarf
- ⇒ **Optimierung** von Requirements-Prozessen
- ⇒ Evaluierung von **Requirements-Tools**
- ⇒ Unterstützung bei Ausschreibungen und **Vertragsverhandlungen**
- ⇒ Projektbegleitendes **Qualitätsmanagement** und **operatives Projekt-Controlling**