Rollen im agilen Requirements Engineering

„Ich bin jetzt Product Owner für unser erstes Scrum Projekt. Da muss ich ja nichts mehr spezifizieren, oder?“ Agil bedeutet ganz und gar nicht, dass von nun an nichts mehr spezifiziert wird! Ganz im Gegenteil. In agilen Projekten wird sehr viel spezifiziert. Nur eben anders und zu anderen Zeitpunkten. Oberstes Ziel ist immer, den Wert der fertigen Software für den Kunden zu maximieren. Genau auf dieses Ziel hin ist das agile Requirements Engineering ausgerichtet.

Agiles RE? Los geht’s!

Agile Methoden, wie z.B. Scrum, ermöglichen es, flexibel auf sich ändernde Kundenwünsche einzugehen und jederzeit neue Themen aufgreifen und umsetzen zu können. Ohne Chaos. Ohne Stress. Und vor allem ohne Qualitätsverlust. Dafür arbeitet das Team das ganze Projekt über eng mit dem Kunden zusammen. So erlauben wir unseren Kunden jederzeit neue Funktion, Änderungen an den Masken, zusätzliche Daten an den Schnittstellen usw. einzukippen. Wir alle lernen im Laufe des Projektes dazu. Das ist auch kein Problem, wenn unsere Methoden, vor allem in Requirements Engineering, entsprechend darauf eingerichtet sind.

Dafür gibt es ganz eigene Techniken zur Spezifikation von Anforderungen. Die klassische Spezifikation in Word wird in agilen Projekten durch schlanke Epics, User-Stories oder Use-Cases abgelöst. Hunderte Seiten dicke Word Dokumente und inkonsistente Excel Listen werden ersetzt durch ein einfach zu handhabendes Product Backlog.

Planung und Priorisierung sind in agilen Projekten sehr wichtig. Anstatt des ohnehin nie aktuellen Gantt- Projektplans gibt es ein Product- und Iteration Backlog. Der Projekt-Controllingbericht zur Verfolgung des Fortschritts der Umsetzung der Anforderungen wird abgelöst von Taskboards und Burndown Charts. Ziel ist es, mit möglichst einfachen Mitteln den Fortschritt im Team zu visualisieren und transparent zu machen.

Agile Methoden haben auch ganz eigene Rollen, die an der Erhebung, Analyse und Umsetzung der Anforde-rungen beteiligt sind. Jede dieser Rollen hat ihre eigenen Aufgaben und Verantwortlichkeiten und erfordert spezielle Fähigkeiten. Und gerade agile Methoden fordern von allen Beteiligten ein hohes Maß an Strukturiertheit und Disziplin!

Was macht agiles Requirements Engineering aus?

In der klassischen Softwareentwicklung nach V-Modell oder Wasserfall werden die Requirements-Engineering-Aufwände sehr stark am Anfang konzentriert und in weiterer Folge noch ein gewisser Teil für das Änderungswesen aufgewendet. Insgesamt geht man von einem empfohlenen Aufwandsanteil von 15-20% des gesamten Projektaufwandes aus (in kritischen Projekten auch höher), um das Requirements Engineering auf einem angemessenem Niveau zu betreiben. In vielen klassischen Projekten wird diese Empfehlung leider ignoriert und der Aufwand für Requirements Engineering oft auf unter 5% gedrückt. Das ist eine der Hauptursachen vieler Projektmisserfolge.

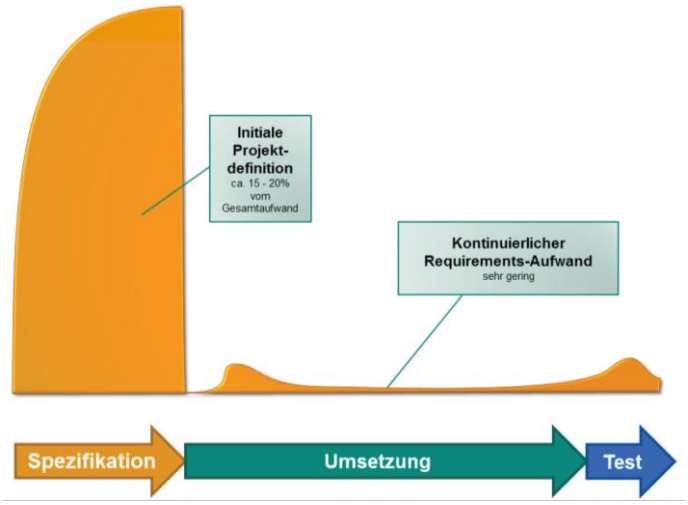


Abbildung 1 - Requirements-Aufwandsverteilung in klassischen Projekten

In agilen Projekten verteilt sich der Requirements-Aufwand in der Regel anders: Man geht von einer geringfügigen Spitze am Beginn des Projekts für Projektsetup und Definition der groben Anforderungen aus. Oft werden je nach Größe des Vorhabens ein bis drei Startiterationen angesetzt, die zu einem guten Teil für Anforderungsklärung verwendet werden. Um nicht in das klassische Schema „möglichst alles vorab spezifizieren“ zu kommen, sollte der initiale Aufwand für Requirements Engineering und Projekt- Setup bewusst niedrig gehalten werden und einen Anteil von ca. 5-10% des Gesamtaufwandes nicht übersteigen.

Der restliche Requirements-Aufwand ist über alle folgenden Iterationen gesehen ziemlich gleichverteilt, da in jedem Sprint Definition, Klärung und Änderung von Anforderungen stattfinden. Hier ist sowohl die Analyse und Spezifikation der Anforderungen für den bzw. die kommenden Sprints gemeint als auch die Requirements-Aufwände im Sprint Planning sowie die dann im laufenden Sprint selbst stattfindende Kommunikation zwischen Product Owner und Team zur

Klärung der vielen auftauchenden Fragen. Es wird geschätzt, dass sich das gesamte Team ca. 10% seiner Zeit über die gesamte Projektlaufzeit mit der Analyse neuer und geänderter Anforderungen beschäftigt.

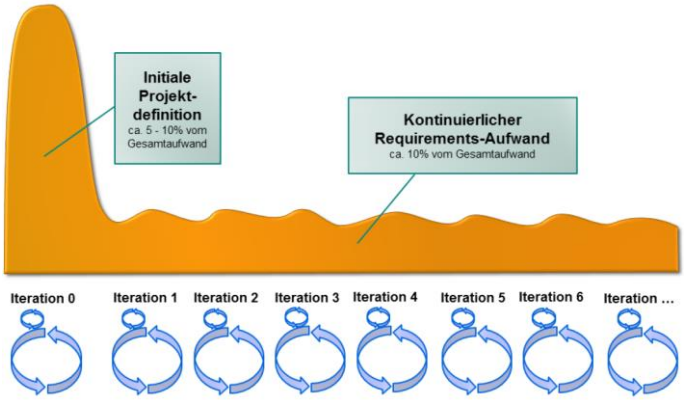


Abbildung 2 - Requirements-Aufwandsverteilung in agilen Projekten

Auch in agilen Vorgehensweisen kann man also davon ausgehen, dass man durch die Anfangsiterationen und die kontinuierlichen Aufwände für die Requirements-Themen über die gesamte Projektlaufzeit auf einen Aufwand von ca. 15-20% kommt. Das ist in etwa gleich oder etwas geringer als in klassischem Vorgehen empfohlen. Der wesentliche Unterschied ist, dass die Aufwände anders verteilt sind und genau dort anfallen, wo sie den größten Nutzen haben.

Beteiligte am Requirements Engineering

Ein wesentlicher Erfolgsfaktor für gut laufende Projekte ist die intensive Einbindung und Kommunikation mit den Fachleuten und Nichttechnikern im Requirements Engineering während des gesamten Projekts. Immer sind unterschiedliche Personen im Fachbereich des Kunden und im Team an der Erhebung, Spezifikation und Ausarbeitung von Anforderungen beteiligt.

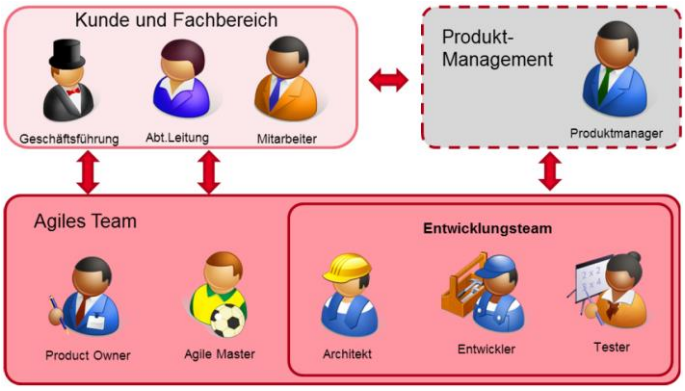


Abbildung 3 - Unterschiedliche Rollen im Fachbereich und im Entwicklungsteam

Abbildung 3 gibt einen Überblick über die beteiligten Rollen im Kunden- bzw. Fachbereich und im agilen Team.

Der Product Owner als Stellvertreter des Kunden

Der Product Owner (PO) ist für die Anforderungen im Product Backlog und deren Priorisierung verantwortlich. Er schreibt die Anforderungen und trifft als deren „Besitzer" alle inhaltlichen Entscheidungen. Er sorgt dafür, dass ständig ausreichend Requirements bereit für die Umsetzung sind und das Team kontinuierlich arbeiten kann. Vor jedem Sprint reiht er die Stories im Backlog und bereitet diejenigen, die im nächsten Sprint umgesetzt werden sollen, vor. Dazu diskutiert er den Inhalt der Story mit dem Team genauer und legt die Akzeptanzkriterien fest.

Ist eine Story fertig umgesetzt, nimmt der Product Owner sie ab. Dies geschieht im Laufe des Sprints sobald eine Story fertig ist, spätestens jedoch im Review Meeting. In diesem stellt das Team die fertigen Stories vor und der PO akzeptiert diese als fertig. Oder auch nicht, wenn noch Mängel auftauchen. Sieht sich der Product Owner die fertigen Stories schon während des Sprints an und führt Tests durch, so bedeutet dies früheres Feedback und weniger Stress am Ende des Sprints.

Als Kunde bzw. Kundenstellvertreter muss der Product Owner die fachliche Welt des Kunden, dessen Ziele und Bedürfnisse sehr genau kennen. Dies ermöglicht es ihm, Entscheidungen für und im Sinne des Kunden treffen zu können. Dies macht die Kommunikation rund um Anforderungen sehr schnell und effizient. Lediglich wenn der Product Owner ein Detail nicht mehr kennt und somit eine Frage nicht mehr selbst beantworten kann, muss er andere Personen (beim Kunden) fragen.

Der Product Owner sollte ständig für das Team verfügbar sein, um Fragen zu beantworten und Feedback zu geben. Anforderungen werden so spät wie möglich spezifiziert. Viele Entscheidungen zu den Anforderungen müssen so erst kurz vor oder während der Umsetzung getroffen werden. Der PO muss während des gesamten Projektes an den Anforderungen arbeiten und für die Klärung von Fragen da sein.

***Beispiel**: In einem plangetriebenen Projekt, beispielsweise zur Erstellung einer Adressverwaltung, würde vor Beginn der Umsetzung exakt beschrieben werden, welche Attribute das Datenobjekt „Adresse" besitzt und welche Datentypen, maximale Länge etc. diese Attribute aufweisen. In agilen Projekten wird vorab lediglich angemerkt, dass es ein Objekt „Adresse" geben soll, eventuell noch ergänzt um die wichtigsten Attribute. Natürlich muss auch in agilen Projekten irgendwann definitiv entschieden werden, welche Attribute das Datenobjekt hat. Dies geschieht hier aber eben nicht schon*

weit vorher, sondern knapp vor oder während der Umsetzung im Sprint gemeinsam zwischen Entwickler und PO.

Der Product Owner hat in agilen Projekten nicht weniger Arbeit mit der Spezifikation der Anforderungen. Der Aufwand verlagert sich lediglich von einem großen Block am Beginn des Projektes hin zu laufenden kleinen Spezifikationsaufgaben und Feedbackzyklen während der gesamten Entwicklungszeit. Die ständige Verfügbarkeit des Product Owner ist deshalb ein kritischer Erfolgsfaktor.

Des Weiteren muss der Product Owner die Anforderungen weniger schriftlich beschreiben, sondern es wird mehr in mündlichen Diskussionen mit dem Team spezifiziert und sofort umgesetzt.

Problematisch: Der Teilzeit Product Owner

Der Teilzeit Product Owner hat eigentlich ganz anderes zu tun. Oft musste er die Rolle des PO zusätzlich zu seinen normalen Aufgaben im Unternehmen übernehmen. Da der Teilzeit Product Owner mit seinem Tagesgeschäft schon ausgelastet ist, hat er kaum Zeit für das Team. Er schafft es gerade noch, z.B. jede zweite Woche einen halben Tag für das grobe Ausarbeiten von Anforderungen aufzuwenden und die wichtigsten Fragen des Teams zu beantworten. Alles andere muss warten. Dass Product Owner zu wenig Zeit haben, ist eines der häufigsten Probleme in agilen Teams und führt zu ernsthaften Schwierigkeiten. Meist liegt die Ursache nicht beim PO selbst, der ja ebenfalls bemüht ist, gute Arbeit zu leisten. Er hat einfach aufgrund seiner Arbeitssituation keine Zeit.

Hier ist der Agile Master gefordert, umgehend eine Lösung gemeinsam mit dem PO und dessen Vorgesetzten zu finden. Fallweise kann ein fähiges Team sicher auch Aufgaben des Product Owners übernehmen, jedoch darf dieser seine Rolle nicht auf Dauer vernachlässigen.

Das Entwicklungsteam als Umsetzer und Berater des PO

Das agile Entwicklungsteam setzt die Anforderungen um. Dabei trifft das Team alle die Umsetzung betreffenden Entscheidungen selbständig. Das Team hat also für Themen wie Architektur, Technologien, Frameworks, Werkzeuge etc. die volle Verantwortung.

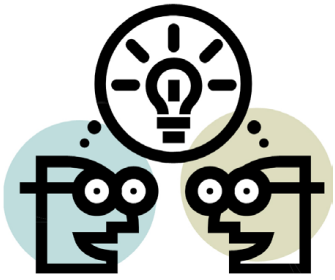
Das Entwicklungsteam prüft und validiert die Anforderungen gemeinsam mit dem PO. Wie in vielen Software Projekten üblich, sind auch in agilen Projekten die Entwickler ein wesentliches Element in der Qualitätssicherung der Anforderungen. Gerade Lücken und Widersprüche in den Anforderungen werden oft erst erkannt, wenn der Entwickler tatsächlich den Quellcode dafür schreibt. Spätestens hier wird sichtbar, dass z.B. für eine „if"-Entscheidung kein „else" spezifiziert ist. Je besser das Team die Welt des Kunden und seine Bedürfnisse kennen, desto besser wird es als zusätzliche Validierungsinstanz wirken können.

Der PO darf sich jedoch nicht darauf verlassen, dass das Team fehlende Teile und Widersprüche in den Anforderungen schon finden wird und er entsprechend weniger Zeit und Energie in die Spezifikation stecken kann. Dies funktioniert nur, solange sehr gute Entwickler im Team sind, die den Fachbereich des Kunden gut kennen und nicht dazu neigen, implizite Annahmen und Entscheidungen zu treffen. Sind diese Punkte nicht gegeben, werden unweigerlich Anforderungen falsch oder unvollständig umgesetzt. Nicht immer sind Fehler in Anforderungen erkennbar. Zum Teil werden Lücken auch instinktiv mit Annahmen geschlossen und der Entwickler kommt gar nicht auf die Idee, nachzufragen. Hier die Schuld beim PO zu suchen, der zu wenig spezifiziert hat, oder beim Entwickler, der nicht nachgefragt hat, ist nicht zielführend.

Ein wichtiges Prinzip im Requirements Engineering lautet daher, dass der PO alles explizit ausspricht, was ihm wichtig ist. Das Team wiederum fragt so lange nach, bis alle Unklarheiten für die Umsetzung beseitigt sind. Da dies stark von Personen abhängt, ist es wichtig, entsprechende Bewusstseinsbildung zu betreiben.

Hier hilft es, wenn der Agile Master in den Sprint Planning Meetings dieses Problemfeld im Auge behält und das Team dazu anleitet, die Anforderungen schon am Beginn des Sprints auf Vollständigkeit, Verständlichkeit, Widerspruchsfreiheit, etc. zu prüfen.

Das Team ist eine Quelle von kreativen Ideen, was mit wenig Zusatzaufwand noch alles möglich wäre. Als Gruppe von Technologieexperten kennt das Team die Möglichkeiten, die das gewählte Lösungsframework bietet. Oft kommen geniale Ideen aus dem Team, was basierend auf dem Kontext des Kunden und den technischen Möglichkeiten noch realisiert werden könnte. In solchen Fällen arbeitet das Team eng mit dem PO zusammen, um die neuen Ideen vorzustellen. Nicht selten kommen aus solchen Ideen Funktionen heraus, die beim Kunden große Begeisterung hervorrufen, da er sich selbst nicht vorstellen konnte, dass dies auch noch realisiert werden könnte.



Das Team schätzt Größe oder Aufwand der Anforderungen im Backlog. Ein wesentlicher Einflussfaktor auf die Sprint Planung ist die Größe bzw. der Aufwand für die Umsetzung einer Anforderung. Das Team hat die Aufgabe, für alle Backlog Elemente die Größe bzw. den Aufwand – je nachdem, welche Art aus Sicht des Teams bzw. im Gesamtkontext gewählt wurde –

zu schätzen. Der Product Owner muss diese Schätzung des Teams respektieren und darf nicht beginnen, die Schätzung nach unten zu verhandeln. Selbst wenn sich das Team zu einer geringeren Zahl überreden lässt, wird am Ende de r tatsächliche Aufwand näher beim ersten, größeren Wert liegen. Planungsfehler sind dann vorprogrammiert.

Das Team berät den PO, wie Anforderungen angepasst werden könnten, um sie doch noch umzusetzen. In vielen Fällen treibt ein kleines Detail an einer Anforderung die Größe bzw. den Aufwand für die Umsetzung extrem in die Höhe. Dies zu analysieren ist Aufgabe des Teams im Zuge der Schätzung. Werden solche Punkte gefunden, berät das Team den PO, wie man die Anforderung abändern müsste, sodass sie mit weniger Aufwand umsetzbar ist. Die letzte Entscheidung hat, wie bei allen inhaltlichen Fragen zu Anforderungen, der Product Owner.

Problematisch: Der Hacker

Der Hacker würde am liebsten nach fünf Minuten Planungsmeeting gleich beginnen zu programmieren. Erlaubt man ihm, ein Notebook ins Meeting mitzunehmen, macht er dies auch schon während des Meetings. Während der Umsetzung will er einfach vorankommen, von Fragen und unvollständigen Anforderungen lässt er sich nicht aufhalten, Lücken füllt er mit aus seiner Sicht vernünftigen Annahmen.

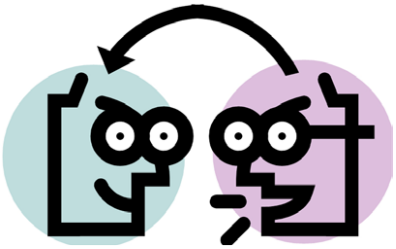
Mut zum Risiko und sich auch mal trauen, mit wenig Informationen zu starten, kann gerade in agilen Projekten sehr wertvoll sein. Nur selten sind Anforderungen soweit ausspezifiziert, dass man wirklich alle Details absehen kann. Dies ist für den Hacker kein Problem. Problematisch wird es, wenn er falsche Annahmen trifft und die Umsetzung dann nichts mehr mit dem zu tun hat, was der Kunde tatsächlich braucht. Dies erkennt ein aktiver PO durch die zahlreichen Feedback Zyklen meist recht schnell. Hier heißt es, gemeinsam mit dem Agile Master gegensteuern und mit dem Hacker vorab genauer durchsprechen, was gefordert ist.

Der Agile Master als Coach und Problemlöser

Der Agile Master ist für den agilen Prozess verantwortlich. Er achtet darauf, dass die gemeinsam festgelegten Regeln eingehalten werden und stößt Anpassungen des Prozesses an. Er schützt das Team vor Störungen von außen und kümmert sich darum, dass es effizient und effektiv arbeiten kann. Der Agile Master sieht sich idealerweise als ein Moderator bzw. Coach und nicht als weisungsbefugter Manager. Er kann von seinem Aufgabenfokus ähnlich einem operativen Prozess- und Qualitätsmanager in herkömmlichen Organisationen gesehen werden. Seine Kernaufgabe liegt darin, dem Team zu helfen, erfolgreich zu arbeiten. Keinesfalls ist der Agile Master verantwortlich für inhaltliche Themen. Dies ist Aufgabe des PO.

Im Requirements Engineering kümmert sich der Agile Master um die Optimierung der Abläufe. Fachlich oder technisch hat der Agile Master im Requirements Engineering keine Aufgaben. Da er kein Projektleiter ist, ist er weder für den Inhalt noch für die erfolgreiche Umsetzung der Anforderungen verantwortlich. Bei allen Meetings und während des Sprints achtet er darauf, dass alle vereinbarten Prozessschritte eingehalten werden, und fordert dies gegebenenfalls ein. Er sieht zu, dass der Product Owner seinen Aufgaben nachkommt, Anforderungen spezifiziert und für das Team da ist. Er stellt sicher, dass das Team eng mit dem PO zusammenarbeitet, bezüglich der Anforderungen ausreichend kommuniziert, diese hinterfragt und laufend umsetzt. Falls das Team mit einem Aspekt des Requirements Engineering nicht zufrieden ist, hilft der Agile Master als Prozessberater dem Team, dies zu verbessern.

Der Agile Master räumt Hindernisse aus dem Weg und schützt das Team. Im Anforderungsmanagement treten immer wieder Hindernisse auf. Es kann zum Beispiel sein, dass der Product Owner zu wenig Zeit hat, Termine kollidieren, Arbeitsmittel wie Flipcharts, Prototyping- oder Modellierungstools fehlen und Ähnliches mehr. Der Agile Master notiert diese Hindernisse im Daily Standup Meeting und beseitigt sie. Neben dem Ausräumen von Hindernissen hat der Agile Master die Aufgabe, das Team vor Änderungen an den für den aktuellen Sprint festgelegten Anforderungen oder neuen Anforderungen während des laufenden Sprints zu schützen. In diesem Punkt muss er auch vom Product Owner Disziplin einfordern und Änderungen und neue Anforderungen bis zum nächsten Sprint zurückstellen.



Der Agile Master ist Coach und Psychologe für das Team. Neben den rein formalen Prozessaspekten beachtet der Agile Master auch das Klima im Team. Gibt es Konflikte, so löst er diese umgehend. Er sorgt dafür, dass die Kommunikation offen und reibungslos verläuft und unterstützt im Umgang mit schwierigen Situationen.

Problematisch: Der geheime Projektleiter

Der geheime Projektleiter weiß zwar, dass er in einem agilen Team arbeitet, insgeheim sieht er aber die ganze agile Vorgehensweise sehr kritisch. Er findet alles chaotisch und unstrukturiert. Die erste kleine Schwierigkeit nimmt er zum Anlass, das Ruder an sich zu reißen und endlich Führung und Linie ins Team zu bringen. Von nun an fühlt er sich verantwortlich für den Fortschritt und treibt das Team vor sich her. Werden Entscheidungen nicht schnell genug vom Team getroffen, so entscheidet eben er. Oft waren geheime Projektleiter früher Projektleiter in klassischen Projekten.

Der geheime Projektleiter nimmt seine Rolle sehr ernst und möchte einen wertvollen Beitrag zum Projekterfolg leisten. Diese Energie sollte für das Projekt auch genutzt werden. Dennoch verletzt er durch sein Verhalten wichtige agile Prinzipien, indem er z.B. verhindert, dass sich das Team selbst organisieren kann. Dies muss so schnell wie möglich angesprochen und gelöst werden. Der Agile Master ist kein Projektleiter, sondern ein Moderator, der das Team zu Entscheidungen motiviert und es schlussendlich zum Projekterfolg begleitet, aber nicht führt.

Der Tester als Prüfer und Qualitätsberater

Der Tester ist von Anfang an im Projekt dabei. Der Tester kann ein Mitglied des Teams oder in einem eigenen Testteam ausgelagert sein. Auch in agilen Vorgehensweisen kann es sinnvoll sein, ein eigenes Testteam zu bilden z.B. für übergeordnete Aspekte wie Integrations- und Systemtests. In jedem Fall ist es sehr sinnvoll, den Tester von Beginn an einzubinden. Anforderungen sind die zentrale Grundlage für die Prüfung der Richtigkeit der Umsetzung, daher muss der Tester diese sehr genau kennen. Dies erreicht man am leichtesten, wenn der Tester von Beginn an bei allen Meetings dabei ist.

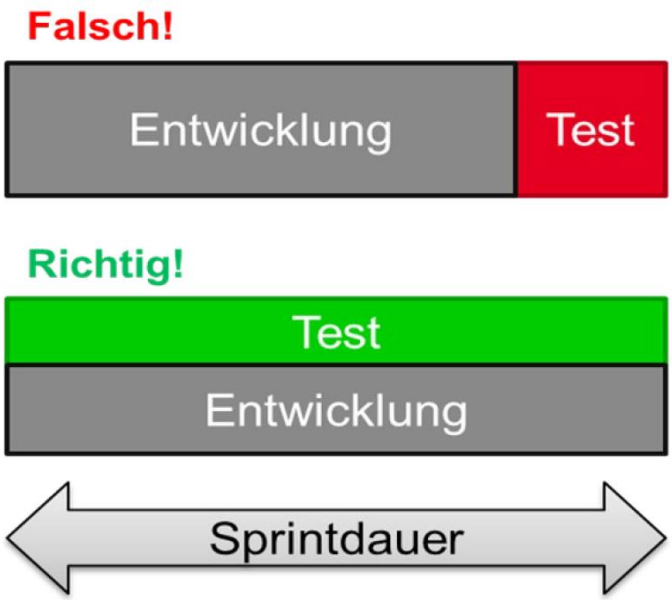


Abbildung 4 - Testen und QS sind von Beginn an ins agile Projekt eingebunden

Der Tester prüft Anforderungen vor der Umsetzung. Im Zuge dieser Vorbereitungsarbeiten des Sprint Plannings beleuchtet der Tester die Anforderungen vor allem im Hinblick auf Verständlichkeit und Testbarkeit. Gerade Tester sind meist sehr gut darin ausgebildet, Fehler zu erkennen und Widersprüche aufzudecken. Diese Fähigkeit ist auch auf Stories und Anforderungen angewandt sehr nützlich. Zusätzlich unterstützt der Tester den Product Owner bei der Definition der Akzeptanzkriterien, die ja die Basis für die Tests und Abnahme der Stories sind.

Der Tester bringt den Bereich Testen und Qualitätssicherung in die Aufwandsschätzung ein. Bei der Schätzung müssen alle Aspekte berücksichtigt werden, die die Größe und den Aufwand beeinflussen. Dazu gehören auch alle Aufgaben rund ums Testen. Der Tester bringt hier sein Know-how ein und berücksichtigt die Testplanung, Testautomatisierung und Testausführung für jede Story. Z.B. kann es bei sicherheitskritischen Stories durchaus vorkommen, dass das Testen mehr Aufwand verursacht als die eigentliche Umsetzung der Story.

Der Tester prüft die korrekte Umsetzung der Stories. Ist eine Story fertig programmiert, prüft der Tester, ob die festgeschriebenen Anforderungen korrekt umgesetzt wurden. Quelle für die Prüfung ist hierbei die Anforderung selbst und die definierten Akzeptanzkriterien. In agilen Projekten ist durch die kurzen Iterationen ein hoher Automatisierungsgrad von Tests erforderlich, entsprechend automatisiert auch der Tester wo immer sinnvoll möglich alle von ihm durchgeführten Tests zur Prüfung der Anforderungen.

Problematisch: Der Perfektionist

Der Perfektionist duldet keinerlei Fehler oder Unschärfen. Alles muss exakt spezifiziert, exakt umgesetzt und bis ins Kleinste getestet sein. Es geht ja schließlich darum, perfekte Qualität abzuliefern. Gerade mit den zu Beginn noch sehr groben Anforderungen hat er große Probleme.

Qualität ist in allen Projekten sehr wichtig, daher ist ein Perfektionist als Tester im Team zwar schwierig, aber möglicherweise auch ein großer Vorteil. Wiederum ist es der Agile Master, der einen möglichen Konflikt erkennen und gegensteuern muss. Vielleicht hilft eine Schulung, in der der Perfektionist lernt, dass man auch qualitativ hochwertige Software hervorbringen kann, ohne alles vorab perfekt zu spezifizieren.

Der Architekt als Berater für das Gesamtsystem

Im Idealfall ist der Architekt Teil des agilen Teams. Es ist jedoch abhängig von der Größe der Organisation bzw. des Produkts eventuell erforderlich, einen teamübergreifenden Architekten zu definieren.

Der Architekt entwickelt eine Gesamtarchitektur, die alle bekannten Anforderungen abdeckt. Zu Beginn der Implementierungsarbeiten wird eine möglichst gute und flexible Architektur entworfen, die zumindest die für die nächsten Iterationen vorgesehenen Anforderungen abdeckt. Je aufwendiger und kostspieliger ein späteres Refactoring ist,

desto mehr Aufwand sollte auch in die Planung einer tragfähigen Architektur investiert werden. In manchen Fällen ist es wirtschaftlich gar nicht mehr vertretbar, in späteren Iterationen große Umbauarbeiten an der Architektur durchzuführen. Architekt, Product Owner und Entwickler im Team arbeiten eng zusammen, um eine optimale Architektur zu finden.

Der Architekt berät den Product Owner, welche Änderungen an den Anforderungen die Architektur vereinfachen und dadurch den Aufwand senken könnten. Ähnlich wie das Entwicklungsteam berät auch der Architekt den PO, welche Details an einer Anforderung die Komplexität der Architektur erhöhen und wie diese Details abgeändert werden könnten, um Komplexität und Aufwand zu reduzieren. Die letzte Entscheidung liegt natürlich beim Product Owner.

Der Architekt schätzt ab, welche Auswirkungen eine Anforderung auf die bestehende Gesamtarchitektur hat. Kommen zu einem bestehenden System neue Anforderungen dazu, so prüft der Architekt, wie diese am besten in der bestehenden Architektur umgesetzt werden können und ob Änderungen an der Architektur notwendig sind. Gemeinsam mit dem Team lässt er dieses Wissen dann in die Größen- bzw. Aufwandsschätzung der Anforderungen einfließen.

Problematisch: Der Generalisierer

Der Generalisierer sucht für alles eine allgemeine Lösung, in die auch zukünftig sicher oder zumindest vielleicht irgendwann kommende Anforderungen schon miteingeplant sind. Auch für das spezifischste Kundenproblem wird eine allgemeine Lösung gesucht, es könnte ja sein, dass man das irgendwann doch einmal umstellen oder für einen anderen Kunden brauchen könnte.

Der Generalisierer argumentiert dabei, dass man oft mit nur wenig Mehraufwand eine allgemeine Lösung umsetzen könne und sich dann in der Zukunft viel Aufwand sparen würde. Dies ist grundsätzlich richtig, allerdings zeigt die Praxis oft, dass entweder eine Notwendigkeit zur erneuten Verwendung nie eintritt oder die realisierte generelle Lösung doch nicht so verwendet werden kann, wie dies der neue Kunde braucht.

Der Generalisierer kann sehr wertvoll sein, da er hilft, sich aufkommende Änderungen vorzubereiten und diese mit möglichst wenig Aufwand umzusetzen. Dennoch ist hier eine sehr intensive Abstimmung mit dem Product Owner notwendig, wo dies Sinn macht und wo nicht.

Der Produktmanager als Dirigent mehrerer Teams

Abbildung 5 zeigt, wie die Organisation aussehen kann, wenn mehrere Produkte von unterschiedlichen Teams entwickelt werden:

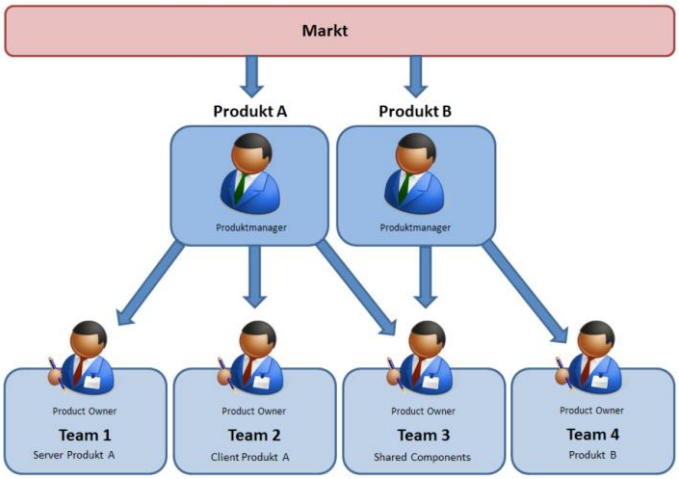


Abbildung 5 - Die Rolle „Produktmanagement“

Der Produktmanager (PM) ist für ein gesamtes Produkt verantwortlich, das von mehreren Teams umgesetzt wird. Die Rolle des Produktmanagers macht immer dann Sinn, wenn mehrere Teams an einem Produkt arbeiten. Der Produktmanager ist dabei nicht Teil eines agilen Teams. Er definiert die Features und Top-Level Anforderungen an ein Produkt. In sehr kleinen Unternehmen, Projekten oder Produktentwicklungen, die mit nur einem einzigen Team arbeiten, wird die Produktmanager und Product Owner Rolle oft von ein und derselben Person wahrgenommen.

Der Produktmanager verwaltet das teamübergreifende Backlog für das Produkt. Der PM definiert die Features für das Produkt im globalen Produkt Backlog. Diese Features werden von mehreren Teams gemeinsam umgesetzt und von den jeweiligen Product Ownern der Teams in Stories heruntergebrochen. Bei der Verwaltung des globalen Backlogs hat der Produktmanager dieselben Kompetenzen und Aufgaben, wie sie der PO für das Backlog des Teams hat.

Der Produktmanager ist nach außen hin ausgerichtet. Während der Product Owner bei größeren Teamstrukturen eng mit dem agilen Team integriert und eher nach innen orientiert ist, ist der Produktmanager klar nach außen, zum Markt oder externen Kunden hin ausgerichtet. Er hat einen guten Überblick über den Markt und die Richtungen, in die sich dieser entwickelt. Er kennt die externen Kunden, deren Wünsche und Bedürfnisse und ist viel auf Messen und direkt bei Kunden unterwegs. Der Produktmanager kennt auch die Konkurrenz sehr gut.

Der Produktmanager plant Releases. Eine zentrale Aufgabe des Produktmanagers ist die Planung von Releases für sein Produkt. Basierend auf den Marktbedürfnissen und den Wünschen der Kunden legt er fest, welche Features und groben Anforderungen in den nächsten Releases umgesetzt werden sollen. Gemeinsam mit den Product Ownern stimmt

er dann diese Wünsche mit den Möglichkeiten der Teams ab und erstellt eine Release Roadmap.

Der Produktmanager arbeitet eng mit den Product Ownern der Teams zusammen. Immer wieder ändern sich die Anforderungen des Marktes, die Wünsche der Kunden und die Prioritäten der bereits geplanten Features und Anforderungen. Der Produktmanager nimmt diese Wünsche und Änderungen auf und stimmt sie mit den Product Ownern ab.

Problematisch: Der „Über den Zaun Werfer"

Der „Über den Zaun Werfer" sieht sich als Auftraggeber und Chef für die Teams. Er sagt, was er möchte, und geht dann davon aus, dass das so gemacht wird. Am liebsten wäre es ihm, wenn das Team erst mit dem fertigen Produkt wieder zu ihm kommt. Ständige Rückfragen und gemeinsames Arbeiten am Produkt sind für ihn unnötig und zeugen nur davon, dass die Entwicklungsmannschaft nicht genug Weitblick und Verständnis für das Produkt hat. Der „Über den Zaun Werfer" nimmt seine Rolle sehr ernst. Er bringt dem Entwicklungsteam viel Vertrauen entgegen und lässt ihm viele Freiheiten, die Details der Lösung zu gestalten.

Meist erkennt der Product Owner als erster, dass die Anforderungen, die er vom Produktmanager bekommt, nicht verwendbar sind. Gemeinsam mit dem Agile Master sollte der PO versuchen, dem Produktmanager klarzumachen, dass es mehr braucht, als zwei visionäre Sätze zu Beginn, um ein Feature gut implementieren zu können. Die Schnittstelle zwischen PO und Produktmanager muss klar definiert werden. Ebenso die Prozesse, wie die Anforderungen gemeinsam erarbeitet werden und welche Qualitätskriterien sie erfüllen müssen.

Literatur & Links

J. Bergsmann, M. Unterauer, Requirements Engineering für die agile Software Entwicklung, 2023, dpunkt Verlag

S. Christmann, Testen in agilen Prozessen an Beispiel Scrum, 2013, Software Quality Lab Knowledge Letter 2013/2 [online]

Autor

Mag. Markus Unterauer

Berater bei Software Quality Lab

Impressum

Der Quality Knowledge Letter ist ein periodisches Informationsmedium von Software Quality Lab und dessen Partnern mit dem Schwerpunkt SW-Qualität.

Weitere Infos zu diesem und anderen Themen finden Sie auf www.software-quality-lab.com

Leistungen von Software Quality Lab

- Tätigkeit als Product Owner und Requirements Engineer in Projekten
- Review und Qualitätsverbesserung von Requirements (User Stories, Use Cases, Prozessbeschreibungen, etc.)
- Einführung und Optimierung von agilem Requirements Engineering in Entwicklungsorganisationen
- Workshops und Coaching für Teams und Product- Owner zum Thema agiles Requirements Engineering
- Seminar „Requirements Engineering für die agile Softwareentwicklung"
- Operative Unterstützung in Projekten und Entwicklungsvorhaben
- Einführung und Anwendung agiler Methoden (Scrum, Kanban)
- Entwickeln eines agilen QS Konzeptes
- Begleitung von großen Softwareprojekten als Pro-zess- und Qualitätsmanager