

Scope Management durch bewusste und transparente Priorisierungsentscheidungen

Priorisierung: Erfolgskritischer Faktor in der agilen Softwareentwicklung

In traditionell geführten Softwareprojekten ist der Scope der zu entwickelnden Software fix, Zeit und Kosten bleiben hingegen variabel. Dies begünstigt u.a. „Scope Creep“, eine schleichende Ausweitung des Umfangs, da Stakeholder im Projektverlauf oftmals zusätzliche Anforderungen zum ursprünglichen Scope stellen.

Scope Creep ist eines von vielen Risiken in klassischen Wasserfallprojekten, denn dadurch wird häufig das geschätzte Budget oder die angenommene Zeitplanung gesprengt – oft zur großen Verwunderung aller Beteiligten. In traditionellen Projekten simulieren Projektpläne den betroffenen Stakeholdern über lange Zeit eine gefühlte Kontrolle (Plan getrieben, vgl. Abb. 1), die sich im weiteren Projektverlauf erfahrungsgemäß nicht bestätigt. Wasserfallprojekte scheitern in der Praxis weitaus häufiger als agil durchgeführte Projekte (s. Abb. 2).

In der agilen Softwareentwicklung sind Zeit und Kosten hingegen fix. Der Umfang wird jedoch geschätzt und bleibt damit flexibel. Beispielsweise entwickelt ein Scrum-Team mit 6 Entwicklern in 2-wöchigen Sprints ein funktionierendes Softwareinkrement, das umgehend Wert für

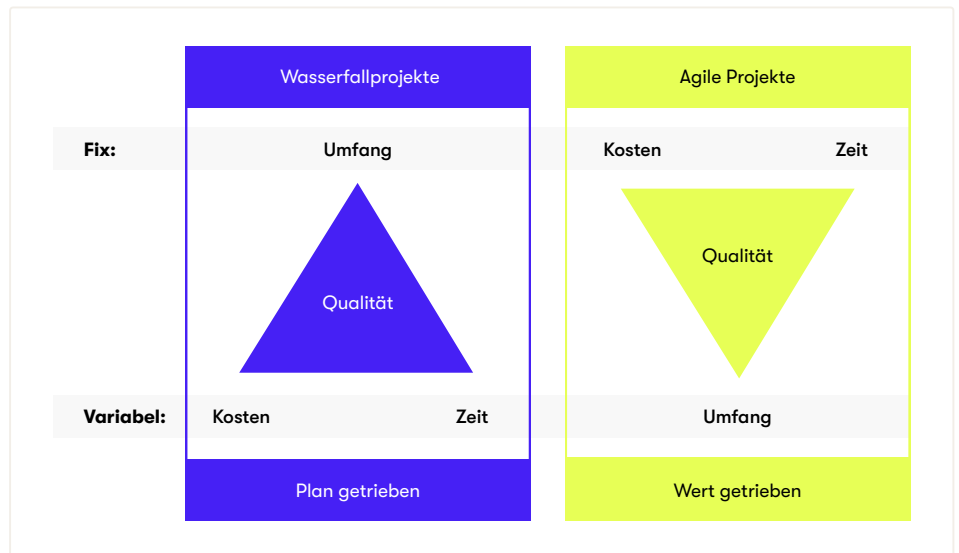


Abbildung 1: Qualitätsdreieck: Wasserfall vs. Agile (scrum.org)

seine Kunden und somit in der Folge für das entwickelnde Unternehmen schafft, sobald es released wird. Im Gegensatz zu traditionell geführten Projekten wird im agilen Kontext der Scope kontinuierlich, immer wieder neu auf genau die Anforderungen ausgerichtet, die den meisten Wert für die Nutzer eines Softwareproduktes generieren (Wert getrieben, s. Abb. 1). Was das Entwicklungsteam in den jeweiligen Entwicklungszyklen umsetzt, liegt dabei in der Verantwortung des Product Owners. Dieser ist dafür verantwortlich, dass das Product Backlog, das eine Auflistung aller zu entwickelnden Anforderungen darstellt, so priorisiert ist, dass genau die Anforderungen als nächstes in die Entwicklung

gehen, die den größten Wert für die Nutzer seines Produktes schaffen. Änderungen von Geschäftsanforderungen, Marktbedingungen oder Technologien sowie identifizierte Risiken im operativen Betrieb des Unternehmens führen immer wieder zu Änderungen des Anforderungsumfangs und dessen Priorität im Product Backlog. Der Product Owner bekommt Anforderungsänderungen oder ergänzende Anforderungen z.B. unmittelbar nach jedem Sprint im Review durch das Feedback der eingeladenen Stakeholder oder nach einem Release durch die Reaktionen der Nutzer seines Produktes zurückgespiegelt. Beim Scrum-Team und den beteiligten Stakeholdern entsteht durch kontinuierliche Feedbackzyklen nach jedem Sprint ein gemeinsames Verständnis darüber, wo die Produktentwicklung zum jeweiligen Zeitpunkt gerade steht und was als nächstes zu tun ist. Dieses gemeinsame Verständnis und die daraus resultierende Transparenz in Bezug auf Entwicklungsthemen ermöglichen dem Product Owner einerseits eine einfachere Kommunikation

Methode	erfolgreich	in Frage gestellt	gescheitert
Agile	39%	52%	9%
Wasserfall	11%	60%	29%

Abbildung 2: Projekterfolg nach Methode (Chaos Report 2015 / The Standish Group)

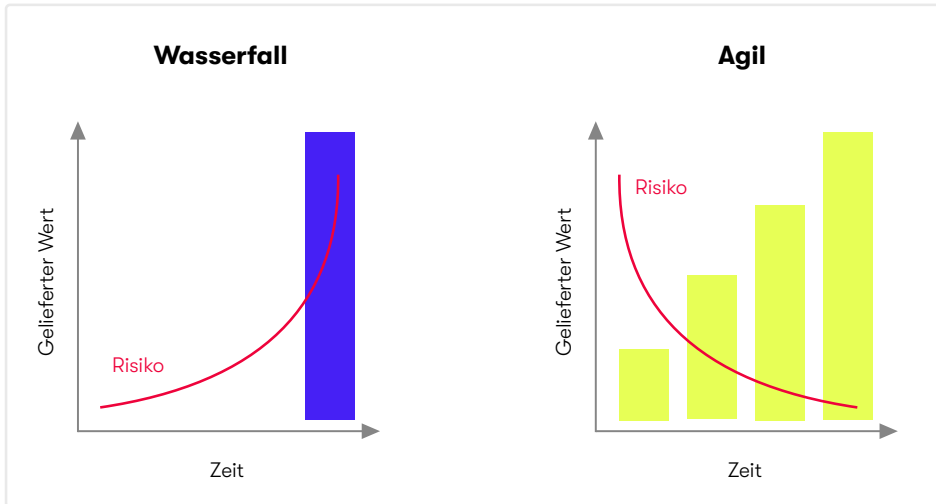


Abbildung 3: Wertgenerierung und Risikomitigierung nach Methode (srum.org)

in Richtungen seiner Stakeholder und dem Entwicklungsteam. Andererseits verhindern sie in der Folge plötzlich auftretende, böse Überraschungen, da entwicklungsbezogene Risiken, Verzögerungen und neue Erkenntnisse bereits frühzeitig gemeinsam mit den Stakeholdern erkannt bzw. an sie kommuniziert werden. Entwicklungsbezogene Transparenz sowie ein gemeinsames, produktbezogenes Verständnis führen ebenfalls zu einem fairen Umgang mit dem Entwicklungsteam. Insbesondere bei Fragestellungen, welcher Scope in welcher Zeit zu schaffen ist, lernen Stakeholdern durch ihre Einbindung in die Feedbackzyklen, was ein Entwicklungsteam leisten kann und was nicht.

In Wasserfallprojekten ist am Projektanfang ebenfalls die Transparenz über den Entwicklungsscope gegeben. Anforderungen werden gemeinsam mit den beteiligten Fachbereichen, die oftmals auch gleichzeitig die Benutzer der Software sind, aufgenommen und in Lastenheften festgehalten. Sobald das Entwicklungsteam damit beginnt diese umzusetzen, verschwindet daraufhin die Transparenz darüber, was es genau entwickelt und wie das Ergebnis aussieht. Nachdem das Entwicklungsteam den definierten Anforderungsumfang nach mehreren Monaten umgesetzt hat, fällt den anforderungstellenden Stakeholdern bei Produktdemonstrationen oder Akzeptanztests auf, dass Anforderungen falsch verstanden und somit falsch umgesetzt wurden oder

in der Zwischenzeit benötigte Funktionen fehlen, weil diese nicht Bestandteil des ursprünglichen Lastenhefts waren. Sie sind u.a. aufgrund von fehlenden Feedbackschleifen unerkannt geblieben. Damit verbunden geht ein Vertrauensverlust der Projektbeteiligten in das gelieferte Softwareprodukt und möglicherweise in das Entwicklungsteam einher. Ebenso entstehen auf Managementebene lange Diskussionen darüber, wer den entstandenen finanziellen Schaden einer nicht bzw. nur teilweise einsetzbaren Software trägt.

Agile Methoden lassen durch die kontinuierliche Priorisierung von Anforderungen jederzeit Anpassungen des Entwicklungsumfangs über den gesamten Produktle-

benszyklus hinweg zu. Sie unterstützen dadurch, dass ein Produkt immer wieder wertschöpfend auf seine Kunden ausgerichtet wird. In der Folge minimiert Agilität so das Risiko, dass ein Produkt an den Bedürfnissen von Nutzern und Markt vorbei entwickelt wird (s. Abb. 3) ohne für diese einen Mehrwert zu bieten. Die Priorisierung des Umfangs bildet dementsprechend einen der wesentlichen erfolgskritischen Faktoren in der agilen Entwicklung, die diese gegenüber traditionell geführten Projekten so erfolgreich werden lässt.

Priorisierung des Entwicklungsumfangs @ nexible

nexible führt die Softwareentwicklung für seine Versicherungsprodukte mit agilen Methoden in einem crossfunktionalen Teamsetup durch. Die einzelnen Teams bestehen einerseits aus Entwicklern, die das eigentliche Softwareprodukt realisieren und andererseits aus Fachexperten diverser Domänen, wie Operations oder Claims, die das Versicherungsprodukt aus fachlicher Sicht aufbauen und betreuen. Sie stellen dem Product Owner beispielsweise ihre fachliche Expertise bei der Anforderungserhebung zur Verfügung.

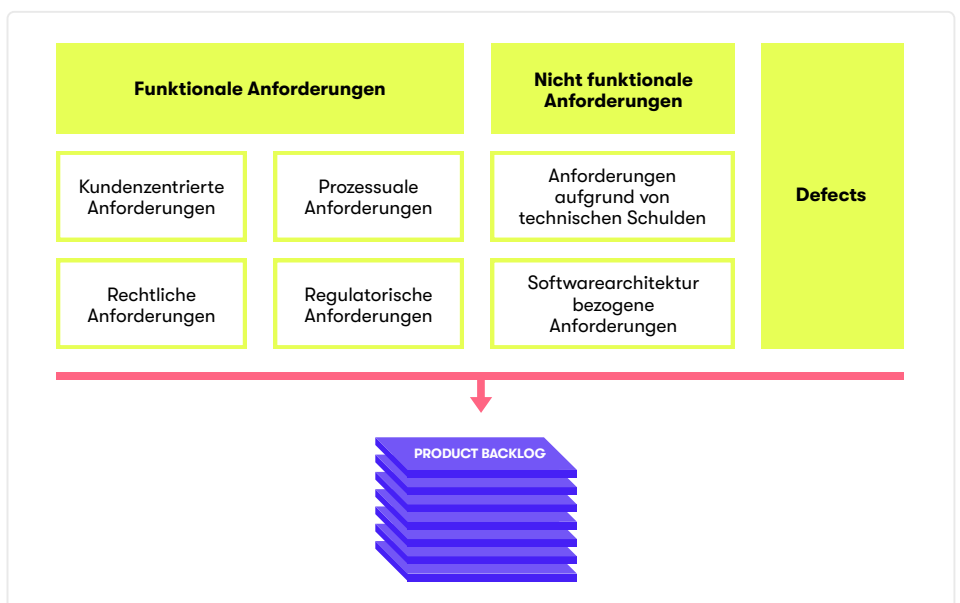


Abbildung 4: Anforderungen in verschiedenen Dimensionen

Das Product Backlog einer digitalen Versicherung setzt sich aus Anforderungen unterschiedlichster Art (s. Abb. 4) zusammen. nexible ist neben kundenzentrierten und prozessualen Anforderungen zusätzlich sehr stark mit regulatorischen und rechtlichen Anforderungen konfrontiert (funktionale Anforderungen). Diese müssen erfüllt werden, um die digitale Versicherungsplattform überhaupt betreiben zu können.

Neben deutlich erkennbaren Anforderungen dürfen bei der Bewertung und Priorisierung des Scopes verdeckte Anforderungen nicht übersehen werden. Dazu zählen beispielsweise Anforderungen, die sich aus technischen Schulden ergeben oder solche, die die Softwarearchitektur betreffen (nichtfunktionale Anforderungen). Zudem kommen Anforderungen in Form von Defects aus dem laufenden Betrieb in das Product Backlog und müssen ggfs. vom Product Owner bei der Priorisierung berücksichtigt werden.

Anforderungen, die bei nexible umgesetzt werden, zahlen immer auf konkrete Unternehmensziele ein. Der Digitalversicherer bewertet die zuvor gesetzte Zielerreichung in jedem Quartal und passt ggfs. die Zielsetzung für das folgende Quartal an. Wie ein Ziel erreicht wird, entscheidet das dafür verantwortliche crossfunktionale

Team jeweils selbst. Es gibt von Unternehmensseite keine Vorgaben auf Initiativen-, Maßnahmen- oder gar Aufgabenebene, was zu tun ist, um das entsprechende Ziel zu erfüllen. Vielmehr fördert der CEO von nexible das eigenverantwortliche Arbeiten der Teams in seiner Rolle als Executive Sponsor und versetzt die Teams in die Lage, eigenständige Entscheidungen treffen zu können.

Beispielsweise erhebt und bewertet der Product Owner für das KFZ-Produkt in Österreich demzufolge nur Anforderungen, die auf ein oder mehrere bei nexible gesetzten Ziele einzahlen. Dies erfolgt ebenfalls quartalsweise nach dem Review der Zielerreichung, gemeinsam mit seinem crossfunktionalen Team in einen Bewertungsworkshop auf Theme bzw. Epic-Ebene. Themes und Epics entsprechen dabei einem sehr groben Anforderungslevel, das bis zur Umsetzung durch das Entwicklungsteam auf ein feingranulareres Anforderungslevel heruntergebrochen wird. Product Owner und Team bewerten im Workshop die identifizierten Anforderungen anhand ihres Impacts in verschiedenen Dimensionen, wie z.B. kommerzieller Impact, Kundenzufriedenheit und Reputation sowie Abhängigkeiten und Deadlines. Sie erarbeiten eine initiale Bewertung anhand diverser Fragestellungen in den zuvor genannten Dimensionen (vgl.

Abb. 5). Diese bildet die Grundlage für weitere Bewertungen des Product Owners, der bestehende Anforderungen bei Unklarheiten ggfs. nach dem Workshop schärft oder sie erneut bewertet, sofern sich neue Erkenntnisse ergeben.

Der Workshop ermöglicht durch die crossfunktionale Zusammensetzung des Teams eine Betrachtung der Anforderungen aus verschiedenen fachlichen und technischen Blickwinkeln. Die dadurch entstehende Bewertungsgrundlage ist durch die „Schwarmintelligenz“ des Teams fundierter, als eine rein durch den Product Owner selbst durchgeführte Bewertung. Der Product Owner bleibt gemäß Scrum Guide jedoch auch weiterhin für alle Anforderungen, die er im Product Backlog aufnimmt und priorisiert, gegenüber den Stakeholdern rechenschaftspflichtig.

Sobald im Workshop eine ausreichend argumentativ gestützte Bewertungsgrundlage vorliegt, vergibt das Team für jedes Epic bzw. Theme jeweils einen initialen Impactwert pro Dimensionen. Die Bewertung erfolgt auf einer Skala von 1 bis 4 (sehr geringer Impact bis sehr hoher Impact in der jeweiligen Dimension). Abb. 6 stellt die schematische Bewertungsmatrix für das KFZ-Versicherungsprodukt in Österreich aus dem ersten Quartal 2019 dar.

	Kommerzieller Impact	Kundenzufriedenheit & Reputation	Abhängigkeiten & Deadlines
Fragen	Wie hoch sind die Abbruchquoten, wenn das Thema nicht umgesetzt wird?	Welche Auswirkungen hat das Thema auf	Gibt es eine gesetzliche Deadline? (z.B. IDD)
	Wieviele Verträge verkauft nexible weniger, wenn das Thema nicht umgesetzt wird?	<ul style="list-style-type: none"> • nexible, • Kunden, • Presse, • Aufsichtsbehörden, • und Social Media, wenn es nicht umgesetzt wird? 	Gibt es eine interne Deadline/Abhängigkeit? (interne Projekte)
	Wieviele Umsatzverlust ist zu erwarten, wenn das Thema nicht umgesetzt wird?		Gibt es eine saisonale Abhängigkeit? (z.B. Wechslergeschäft)

Abbildung 5: Beispiele aus den Fragestellungen pro Bewertungsdimension

Themen & Epics	Anforderungsdimension	Gesamtimpact	Kommerzieller Impact	Kundenzufriedenheit & Reputation	Deadline & Abhängigkeit
Epic 1	Prozessual	11	4	3	4
			Argument 1 Argument 2	Argument 1 Argument 2	Argument 1
Epic 2	Prozessual	11	3	4	4
			Argument 1 Argument 2 Argument 3	Argument 1 Argument 2 Argument 3 Argument 4	Argument 1 Argument 2
Theme 1	Kundenzentriert	8	2	3	3
			Argument 1	Argument 1 Argument 2	Argument 1
Epic 3	Prozessual	6	2	4	0
			Argument 1 Argument 2	Argument 1	Kein Argument

Abbildung 6: Bewertungsmatrix für das KFZ-Versicherungsprodukt Österreich (Q1 2019)

Die Summe der einzelnen Impactwerte pro Dimension ergibt einen Wert für den Gesamtimpact. Dieser ermöglicht es dem Product Owner die Epics und Themes zueinander zu priorisieren. Die Priorität der Anforderungen spiegelt sich in der Reihenfolge im Product Backlog wieder (s. Abb. 7) und ist dort jederzeit für alle beteiligten Stakeholder transparent einsehbar. Sofern im Tagesgeschäft neue Anforderungen erkannt werden, bewertet der Product Owner sie ebenfalls nach der gleichen Methode.

Seit Einführung der Bewertungsworkshops bei nexible, stieg die Transparenz

im Team sowohl über die gesetzten Ziele als auch über die dafür umzusetzenden Anforderungen und deren Impact auf die Zielerreichung signifikant. Dies hatte zur Folge, dass das Team bei anschließenden Refinements, in denen Anforderungen vom Theme- und Epic-Level zu umsetzbaren Entwicklungsaufgaben heruntergebrochen werden, bereits ein besseres Verständnis davon hatte, was zu tun ist und somit die Themen fokussierter anging. Ebenfalls reduzierten sich dadurch Kommunikationsaufwände bei der Erklärung der Anforderungen auf der Seite des Product Owners.

Final betrachtet, ermöglicht eine bewusst

durchgeführte Anforderungsbewertung dem Product Owner eine einfachere Priorisierung des Entwicklungssscopes, um genau die Anforderungen zum richtigen Zeitpunkt in die Entwicklung zu bringen, die den größten Wert für die Nutzer seiner Software generieren. Die Kunst besteht darin, sich schnell von den Anforderungen zu trennen, die keinen oder nur sehr geringen Kundennutzen schaffen. Dies fokussiert den ökonomischen sinnvollen Einsatz der Entwicklungsteams in einem agilen und softwaregetriebenen Versicherungsunternehmen, wie nexible. Ebenfalls unterstützt ein für alle Beteiligten nachvollziehbarer und transparenter Bewertungsprozess den Product Owner in der Argumentation seiner Priorisierungsentscheidungen gegenüber seinen Stakeholdern sowie bei der Kommunikation in seinem Team.

Der Product Owner kann dadurch die beiden Kernaufgaben seiner Rolle, Wertmaximierung des Produktes und Stakeholdermanagement, insgesamt besser erfüllen.



Sascha Auerbach begleitete als Product Owner den Rollout des nexible KFZ-Produkts in Österreich. Er ist Senior Consultant bei CGI im Bereich Insurance.

CGI

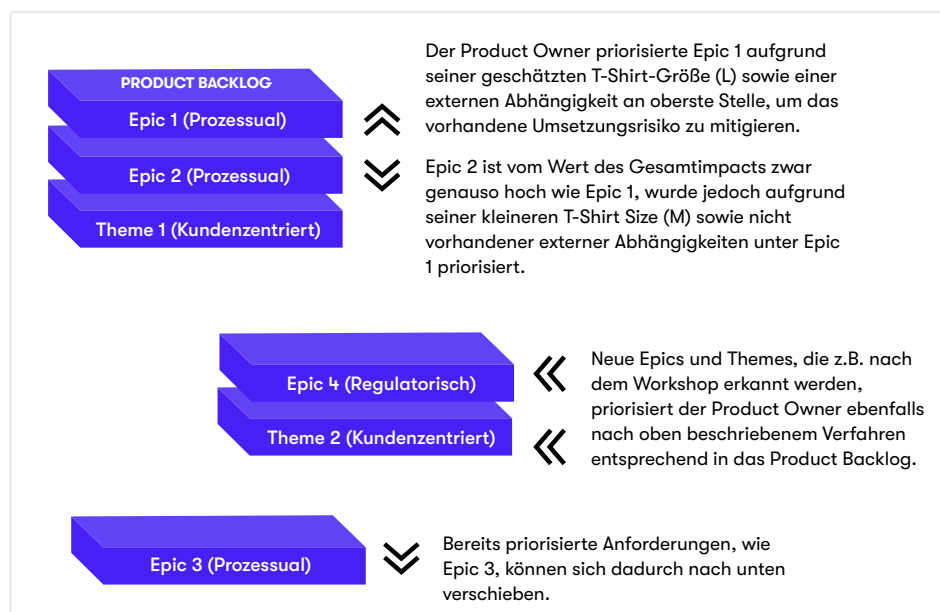


Abbildung 7: Priorisiertes Product Backlog für das KFZ-Versicherungsprodukt Österreich (Q1 2019)