AGIL AM LIMIT

AGIL AM LIMIT

Wie Du mit AgileTrio agile Werte und Vorgehen im streng regulierten Umfeld umsetzen kannst

von Jonathan von Wittern

Impressum:

Lektorat: Julia Auth
Farben & Schrift: Designhoheit (https://designhoheit.de/)

ISBN Hardcover: 978-3-384-08287-9
ISBN E-Book: 978-3-384-08288-6

Druck und Distribution im Auftrag des Autors:

tredition GmbH,
Heinz-Beusen-Stieg 5,
22926 Ahrensburg,
Germany

Inhalt

Vorwort

Liebe Leserin, lieber Leser,

seit über 20 Jahren arbeite ich nun im regulierten Bereich. Begonnen habe ich meine Karriere in der Fertigung Medizinischer Geräte. Dann ging es weiter mit Kundenservice und Reparaturen Medizinischer Geräte. Es folgte eine Zeitspanne in der Entwicklung von Embedded Software für Luft- und Raumfahrt, Automotive und Medizinprodukte. Schließlich fand ich mich im Bereich Quality und Design Assurance der Medizintechnik wieder. Dort hatte ich auch zum ersten Mal eine Führungsposition inne.

Während meiner gesamten beruflichen Laufbahn bewegte ich mich bis dahin in einem traditionellen Arbeitsumfeld und arbeitete vorzugsweise in Projekten. Obwohl agiles Mindset und Methoden bereits vor über 20 Jahren entwickelt und im Laufe der Zeit immer populärer wurden, dauerte es mehr als 15 Jahre, bis ich die Gelegenheit hatte, mich im Detail mit agiler Softwareentwicklung in einem Startup für Medizinische Software auseinanderzusetzen.

Selbstverständlich gab es zuvor Versuche im konservativen Umfeld der Medizintechnik, agil zu arbeiten. Besonders in der Softwareentwicklung war dies am einfachsten umzusetzen. Dennoch waren die Ergebnisse oft enttäuschend. In den meisten streng regulierten Organisationen wurde die Produktentwicklung weiterhin in Form von Projekten durchgeführt, und die Strukturen blieben klassisch. Nur die Softwareentwicklung versuchte, sich agil zu organisieren, oft in Form von Scrum im V-Modell (V-Trichter).

In der Regel blieb es jedoch dabei. Scrum wurde vor allem genutzt, um Softwareteams zu organisieren, aber von echter Agilität konnte keine Rede sein. Die Produktentwicklung dauerte immer noch zwei bis drei Jahre, und Anpassungen am Projekt waren nur über moderate Iterationen eines Stage-Gate Prozesses möglich.

Eine andere Situation ergab sich in Startups für Medizinische Software. In meiner aktuellen Position im Bereich Regulatory Affairs, mit einem breiten Spektrum an Aufgaben und Verantwortung, arbeite ich in jungen Unternehmen, die sich auf die Digitalisierung des Gesundheitswesens spezialisiert haben. Hier habe ich hauptsächlich mit Medizinischer Software und Digitalen Gesundheitsapplikationen (DIGAs) zu tun.

Am Anfang ist alles oft wild und chaotisch. Die Softwareentwicklung beginnt in der Regel mit einer Mischung aus Scrum und Kanban, was als agiler Ansatz betrachtet werden kann. Doch bei der Marktzulassung gibt es immer wieder unangenehme Überraschungen. Gesetzliche Anforderungen und Standards werden häufig entweder nicht ausreichend berücksichtigt oder komplett ignoriert, was zu schwerwiegenden Verstößen gegen die Bedürfnisse der Nutzenden des Produkts führt.

Selbstverständlich kann man das Scrum Team im agilen Vorgehen immer wieder ausprobieren lassen, die Marktzulassung doch noch irgendwie durch Versuch und Scheitern zu erwirken und das Team daran wachsen lassen. Doch ohne die notwendige Erfahrung, Kenntnisse sowie ohne die Techniken, Methoden und Tools kann dies ein sehr langwieriger Prozess sein, der mit viel Frust und Ressourcenverlust einhergeht.

Darüber hinaus stellt das Scheitern bei der Marktzulassung insbesondere für junge Unternehmen und Startups ein existenzielles Risiko dar. Investoren reagieren empfindlich, wenn die Marktzulassung nicht reibungslos verläuft. Geldgebende verstehen sehr gut, welche Folgen ein gescheitertes Zulassungsverfahren hat und erkennen sofort, dass dies auf mangelnde Beachtung von Gesetzen und Standards zurückzuführen ist. Die Bereitschaft, weiteres Geld zu investieren, schwindet schnell, und das Startup kann in die Insolvenz geraten.

Dieses Buch hilft dir dabei, das Risiko bei Softwareprodukten in streng regulierten Bereichen zu reduzieren. Es richtet sich an ganze Teams, Scrum Master und Product Owner aus der agilen Welt, die Verantwortung für streng regulierte Produkte tragen, sowie an Qualitätsmanager*innen und Regulatory Affairs aus dem klassischen Bereich, die sich mit der agilen Haltung und Vorgehensweise vertraut machen möchten. Auch Topmanager*innen und Geschäftsführende in jungen Organisationen und Startups profitieren von diesem Buch, da es wertvolle Erkenntnisse liefert.

Um beide Welten zu vereinen, behandelt das Buch sowohl klassische als auch agile Ansätze und Denkweisen. Ziel ist es, Angehörigen beider Welten zu vermitteln, wie die jeweils andere Seite funktioniert, welche Vor- und Nachteile beide Welten mit sich bringen und warum beide Ansätze nach wie vor relevant sind. Es ist klug, sich in beiden Welten auszukennen und das Beste aus diesen zu nutzen.

Das Buch widmet sich zudem dem Thema Qualitätsmanagementsystem, da im streng regulierten Umfeld „Zufall und Zuruf“ nicht ausreichen. Dies erfordert eine Umstellung, besonders von der Geschäftsführung und den Process Owner, die ihre Rolle verstehen und konkrete Verantwortung übernehmen müssen.

Das Herzstück meines Buches ist AgileTrio. Hier wird der Produktlebenszyklus für Medizinische Software in Schleifen durchlaufen. Der Ansatz lässt sich jedoch leicht auf andere Softwareprodukte im regulierten Bereich übertragen, indem wenige Anpassungen vorgenommen werden. Das Ziel ist, die Release Zyklen so kurz wie möglich zu halten, wobei „Continuous Documentation“ der Schlüssel ist.

Zudem gehe ich detailliert auf Themen wie die Technische Dokumentation, das Risikomanagement, das Anforderungsmanagement, die Verifizierung, die Validierung und die Softwareproduktion ein. Ich bringe dir Techniken und Methoden näher, die dir helfen sollen, saubere und nutzbare Ergebnisse zu erzielen.

Da heutzutage kein Softwareprodukt mehr ohne Informationssicherheit und Datenschutz auskommt, widme ich auch diesem Thema ein eigenes Kapitel zur Vertiefung. Hier zeige ich, worauf es ankommt und worauf im Detail zu achten ist. Besonders beim Thema Datenschutz kursieren viele Halbwahrheiten. Teilweise sieht man sich mit Frustration und Naivität konfrontiert.

Die Kapitel zu den Themen Überwachung nach dem Inverkehrbringen und Clinical Affairs sind zwar spezifisch für Medizinprodukte, doch besonders die Überwachung nach dem Inverkehrbringen zeigt, dass es sich eigentlich um „Continuous Feedback“ handelt, das für viele Arten von Produkten interessant ist, unabhängig von der Vorgehensweise.

Abschließend richte ich den Fokus auf meinen Spezialbereich - das Regulierte Umfeld. Dieses Kapitel ist insbesondere für diejenigen unter euch von Interesse, die schon immer genau wissen wollten, was Regulatory Affairs eigentlich macht. Auch hierbei nähere ich mich dem Thema anhand Medizinischer Software, gehe aber zusätzlich auf allgemeine Gesetzgebung, vorzugsweise in der Europäischen Union, ein.

Es wäre natürlich wünschenswert, wenn du das komplette Buch lesen würdest. Es ist aber nicht zwingend erforderlich. Wer sich nicht lange mit irgendwelchen Mindsets oder Vorgehensweisen beschäftigen möchte, sondern lieber direkt wissen will, was Sache ist, kann gerne mit dem Kapitel *AgileTrio* beginnen. Es ist nicht zwangsläufig notwendig die Kapitel davor gelesen zu haben. Bei den Vertiefungsthemen, die folgen, steht ebenfalls jedes Kapitel für sich. Ich habe das Buch extra so konzipiert, dass du dich nur mit den Themen auseinandersetzen kannst, die akut sind oder dich persönlich betreffen und ansprechen.

Innerhalb des Buches gibt es immer wieder kleine Exkursionen in Form von gelben Notizboxen, die oft meine persönlichen Erfahrungen und Perspektiven

widerspiegeln, aber die vor allem dich zum Nachdenken anregen sollen. Ich hoffe, du hast Freude daran und es bringt dich gelegentlich zum Schmunzeln.

Die Idee, dieses Buch zu schreiben, kam mir während der Entwicklung und Implementierung des AgileTrio Ansatzes. Diese Zeit war intensiv und erlaubte kaum Fehler. Denn wie schon beschrieben, kann das Scheitern einer Markzulassung in einem Startup das Ende bedeuten. Umso erfreulicher war deswegen die Abnahme der Implementierung des AgileTrio Ansatzes durch eine Benannte Stelle anhand einer Medizinischen Software. Zwar war uns vorher bereits bewusst, dass es für die Scrum Teams funktioniert. Anschließend bekamen wir jedoch zusätzlich die Bestätigung, dass es auch regulatorisch funktioniert.

Ich möchte mich für diese Erfahrung besonders bei zwei Personen bedanken. Vaclav Vlcek, der mir als Qualitätsmanager den Rücken freigehalten hat und es dadurch ermöglicht hat, mir ausreichend kreative Zeit mit den Scrum Teams und der Organisation für die Umsetzung des AgileTrio Ansatzes zu verschaffen. Mein besonderer Dank gilt zudem Anna Winkler, eine Kollegin und Scrum Master, für die zahlreichen Diskussionen und Debatten, ihre unnachgiebige Art, Dinge zu hinterfragen, und ihr großartiges Kommunikationstalent. Ohne diese beiden Personen wäre die Entwicklung und Implementierung des AgileTrio Ansatzes in dieser Form nicht möglich gewesen.

Es hat mehr als ein Jahr gedauert, bis ich schließlich mit dem Schreiben dieses Buches begonnen habe. Das Schreiben selbst hat mir geholfen, sämtliche berufliche Erfahrungen zu reflektieren, zu hinterfragen und auch zu verarbeiten. Dafür blieb zuvor oft nicht die Zeit. Bis zur Veröffentlichung dieser 1. Auflage verging nochmals über ein Jahr. Dabei habe ich gelernt, dass das Schreiben eines Buches einem Marathonlauf ähnelt und enorme Ausdauer und Durchhaltevermögen erfordert.

Mein Dank gilt hier Thomas Kluge für sein fachliches Review, neue Denkanstöße und die Motivation. Selbstverständlich darf auch meine Lektorin Julia Auth nicht unerwähnt bleiben, die nicht nur für sprachliche Korrekturen gesorgt hat, sondern auch darauf geachtet hat, dass alles schlüssig ist und zusammenpasst.

Ein abschließender und ganz besonderer Dank gilt meiner wunderbaren Frau Bernadette von Wittern, für ihre Geduld, die vielen Stunden des Zuhörens und dafür, dass sie mich regelmäßig auf den Boden der Tatsachen zurückgeholt hat. Zudem haben sie und unsere beiden Kinder mir die Zeit eingeräumt, dieses Buch schreiben zu können. Sie alle haben Geduld bewiesen, wenn ich mich nicht sofort vom Schreiben lösen konnte.

Nun möchte ich dich nicht länger auf die Folter spannen und meinen AgileTrio Ansatz und meine Gedanken dazu in dieser 1. Ausgabe mit dir teilen.

Jonathan von Wittern

Auf euer Feedback, eure Anregungen sowie Gedanken freue ich mich immer unter:

AgilamLimit@ProductLounge.net

1. Wie hat sich Softwareentwicklung verändert

Die Softwareentwicklung ist im Vergleich zu anderen Entwicklungsdisziplinen wie Hardware oder Mechanik noch relativ jung. Sie ist jedoch die abstrakteste Disziplin. Software kann nicht angefasst werden und ist oft schwer verständlich für Menschen, die keine Programmierkenntnisse haben. Um als Softwareentwickler*in verständlich zu sein, benötigt man neben Programmierfähigkeiten auch gute Kommunikationsfähigkeiten, um Sachverhalte zu veranschaulichen und zu dokumentieren. Es ist zudem wichtig zu verstehen, dass Software ohne Hardware nicht existieren kann. Dies wird zuweilen übersehen. Wenn es um angemessene Testtiefen geht, verwende ich oft den Spruch „Software läuft nicht in der Luft“.

Die Hardware umfasst physische Komponenten, die benötigt werden, um Software zu speichern und auszuführen. Die Hardware verfügt in der Regel über Schnittstellen, die Ein- und Ausgänge ermöglichen. Diese Schnittstellen können Dateneingänge und -ausgänge sowie Aktoren und Sensoren sein. Die Hardware umfasst auch Speichereinheiten wie Mikrochips und Festplatten, Prozessoren sowie Eingabegeräte wie Maus oder Tastatur.

Die Software besteht aus Verfahren, die in Zusammenarbeit mit der Hardware bestimmte Aufgaben erfüllen. Es handelt sich um geordnete Anweisungen, die Veränderungen in der Hardware bewirken. Dadurch können wir mit der Hardware kommunizieren und interagieren. Menschen tun dies normalerweise über Nutzerschnittstellen. Aber auch andere Hardware oder Sensoren können mit der Hardware interagieren und über geordnete Anweisungen weitere Hardware oder Aktoren ansprechen.

Die Softwareentwicklung begann in den 1950er Jahren, in denen Software und Hardware als unterschiedliche Dinge betrachtet wurden. Es war auch die Zeit der ersten Transistoren, was die Möglichkeit bot, Schaltkreise immer kleiner zu machen. In den 1960er Jahren kamen die ersten Minicomputer auf und die Softwareentwicklung konnte breiter voranschreiten. Gleichzeitig stieg der Bedarf an Software. In den 1970er Jahren wurden mit den Mikrocontrollern Computer immer kleiner und leistungsfähiger.

Jedoch begann 1965 die sogenannte große Software-Krise, die bis etwa 1985 andauerte. In dieser Zeit wurden viele Probleme der Softwareentwicklung identifiziert und bekannt. Budgets und Zeitpläne gerieten regelmäßig außer Kontrolle. Fehlerhafte Software verursachte große Schäden und kostete zum ersten Mal auch Menschenleben. Ein Beispiel dafür sind die Strahlentherapieunfälle Anfang der 1980er Jahre,

bei denen fehlerhafte Software dazu führte, dass Geräte eine tödliche Strahlendosis an Patienten*innen abgaben.

Das Wasserfallmodell entstand in dieser Zeit. 1970 veröffentlichte Winston W. Royce eine erste Publikation darüber. Das Ziel war, große Softwareprojekte besser kontrollieren und fatale Fehler vermeiden zu können. Dadurch sollten Kosten, Zeit und Ressourcen besser planbar und kontrollierbar werden.

Das Requirements-Engineering hat ebenfalls seine Ursprünge in den 1970er Jahren. Das Ziel war, Anwenderbedürfnisse durch systematische Anforderungsanalyse besser zu berücksichtigen und umzusetzen. Das Requirements-Engineering wurde entwickelt, um Projekte, die Kosten, Zeit und Ressourcen außer Kontrolle geraten lassen, zu vermeiden.

In den 1980er Jahren entstanden verschiedene Vorgehensweisen und Lösungen, die als „Silver Bullets“ angepriesen wurden, um die Software-Krise zu bewältigen. Doch in den 1990er Jahren setzte die Ernüchterung ein. Fred Brooks veröffentlichte einen Artikel Mit dem Titel „No Silver Bullet“, in dem er argumentierte, dass keine einzelne Technologie oder Praxis innerhalb von zehn Jahren eine 10-fache Verbesserung der Produktivität bewirken würde.

Mit den 1990er Jahren begann der Aufstieg des Internets, was zu einer großen Nachfrage nach unkritischer und anwendungsbezogener Software führte. Die Softwareentwicklung wurde durch Skriptsprachen erleichtert. In dieser Zeit entstand auch das V-Modell als Weiterentwicklung des Wasserfallmodells, das moderate Iterationen ermöglicht. Das Requirements-Engineering entwickelte sich ebenfalls weiter. Die Unified Modeling Language sollte dabei helfen, Software zu modellieren, anstatt sie selbst zu programmieren. Es bot auch die Möglichkeit, Anforderungen visuell darzustellen, anstatt nur sprachlich.

Anfang der 2000er Jahre wurde die ISO/IEC 25000er Reihe [1] veröffentlicht. Sie stellt ein standardisiertes Werk für Anforderungen, Qualität und Bewertung von Softwareprodukten dar. Diese Anforderungen sind wissenschaftlich fundiert und entsprechen dem „Stand der Technik“. Sie werden auch heute noch in der klassischen Softwareentwicklung angewendet.

Mit dem Aufstieg des Internets und der starken Nachfrage nach anwendungsbezogener und unkritischer Software wuchs auch der Frust bei Teilen der Softwareentwicklung über herkömmliche Methoden. Dadurch entstand bis 2001 das Agile

Manifesto, das als Ursprung und Basis aller nachfolgenden agilen Vorgehensweisen, Prinzipien und Praktiken gilt.

Kanban hat seinen Ursprung bei Toyota und wurde in den 1940er Jahren zunächst in der Autoproduktion eingesetzt. In der Softwareentwicklung wurde es erst etwa 70 Jahre später, um 2010, erstmals angewendet. Kanban gilt als eine der radikalsten agilen Methoden, die jedoch nur mit großer Teamdisziplin gut funktionieren kann.

Scrum dagegen entstand bereits in den 1990er Jahren durch Jeff Sutherland und Ken Schwaber. Es wurde jedoch erst Anfang der 2000er Jahre bekannt, als beide Vordenker es zum ersten Mal publizierten. Scrum ist ein Time-Boxing-Verfahren, das von Natur aus Disziplin erfordert und als das erfolgreichste agile Vorgehensmodell gilt. Auch im agilen Vorgehen ist Anforderungsmanagement wichtig, jedoch oft direkter, leichter und unvollständiger.

Moderne Ansätze für agile Softwareanforderungen versuchen, systematische Ansätze anzubieten, die das Dokumentieren und Spezifizieren nicht überbetonen, aber dennoch eine Mindestspezifikation bieten. Oft wird die Dokumentation auf Organisationsebene auch als Anforderungsdokumentation für ein Produkt genutzt. Diese Ansätze sind relativ neu und weniger als zehn Jahre alt.

Heutzutage stehen wir vor den Herausforderungen der Digitalisierung in einer schnelllebigen Welt. Unsere Nutzung von Cloud-Services vermittelt uns den Eindruck, dass Hardwareressourcen unbegrenzt verfügbar sind. Software spielt eine zentrale Rolle und wird auch in kritischen Bereichen wie autonomen Fahren, Medizinischer Software, dem Internet der Dinge, maschinellem Lernen und künstlicher Intelligenz eingesetzt. Daher besteht ein wachsender Bedarf an agilen Methoden für streng regulierte Produkte.

Mein Ansatz, den ich in diesem Buch erläutere, ist AgileTrio, das Organisationen und Teams dabei helfen kann, regulierte und qualitativ hochwertige Software flexibel, anpassungsfähig und schnell zu entwickeln. Dabei sollen moderne agile Werte nicht vernachlässigt werden. Dazu wird Scrum mit einem regulierten Framework kombiniert. Darauf werde ich im Kapitel *AgileTrio* näher eingehen.