FHS St.Gallen
Hochschule
für Angewandte Wissenschaften

AGILES TESTEN
IN
ERP PROJEKTEN

Verfasser:	Nando Costantino
Referent:	Silvio Moser
Studiengang:	MAS Business Process Engineering
Eingereicht am:	09.03.2020

Vorwort

Das Themengebiet ERP begleitet mich bereits über mehrere Jahre in meinem beruflichen Alltag. Als Unternehmensberater für Evaluationen und Einführungen von ERP-Systemen sowie später als ERP-Systemverantwortlicher bei meinem jetzigen Arbeitgeber konnte ich viel Erfahrung sammeln. Während des Studiums an der Fachhochschule St. Gallen, sowie im beruflichen Alltag wurde Agilität je länger je präsenter. Die Vorteile von agilem Vorgehen waren für mich offensichtlich. Aus diesem Grund wollte ich mehr über Agilität erfahren. Umso mehr erfreute mich die Entscheidung, dass mein jetziger Arbeitgeber die anstehende ERP-Einführung agil umsetzten möchte. Dabei ist agiles Testing ein Themengebiet, bei welchem in meinem beruflichen Umfeld noch wenig Expertise vorhanden ist. So entschied ich mich bei meiner Masterarbeit für das Thema «Agiles Testen in ERP-Projekten».

Mit der Wahl dieses Themas konnte ich mir viel Wissen in Bezug auf Agilität aneignen, und gleichzeitig für mein Arbeitgeber Mehrwert schaffen. Interesse an einem Themengebiet, kombiniert mit einem effektiv vorhandenen Bedarf schafft eine gute Ausgangslage für die Erstellung einer Masterarbeit.

An dieser Stelle möchte ich mich bei allen Personen bedanken, welche mich in dieser anstrengenden Zeit unterstütz haben. Im Speziellen bei meiner zukünftigen Ehefrau Nadja Bechtiger, welche mich stets unterstütz hat und mir den Raum und die Zeit für die Erstellung meiner Arbeit gegeben hat.

Ein spezieller Dank geht an meinen Referenten Silvio Moser. Durch seinen grossen Erfahrungsschatz im agilen Umfeld und auch im Testing habe ich viele wertvolle Tipps für meine Arbeit erhalten. Auch ist es nicht selbstverständlich, dass ein Geschäftsführer eines erfolgreichen Unternehmens sich die Zeit nimmt, Studenten bei der Masterarbeit zu begleiten. Der Aufwand dafür ist nicht unerheblich, was ich sehr zu schätzen weiss.

Zuletzt möchte ich mich noch bei den Fachexperten bedanken, welche sich für die Interviews zur Verfügung gestellt haben. Alle Experten haben mir ohne zu zögern Ihre Bereitschaft zugesichert. Dank diesen Beträgen wurde diese Arbeit erst ermöglicht.

Nun wünsche ich allen, die sich für agiles testen in ERP-Projekten interessieren, viel Spass beim Lesen meiner Ausführungen.

Schänis, März 2020 Nando Costantino

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis und Glossar

Begriff	Erklärung
Agile Coach	Trainer / Mentor im agilen Umfeld
Agiles Manifest	Öffentliche Erklärung / Statement bezüglich agiler Vorgehensweisen
Agilität	Flexibilität, Anpassungsfähigkeit
Akronym	Wort, das gewollt aus der Zusammensetzung von verschiedenen Wörtern entsteht
Anforderungen	Spezifische Ausprägung / Funktion von Software
ATDD	Acceptance Test Driven Development
Branche	Gruppierung / spezifische Ausprägung von Unternehmen
BDD	Behavior Driven Development
Build	Bauen / Erstellen; Software erstellen
Business Analyst/in	Spezialist/in für Analyse und Dokumentation von Unternehmensprozessen
Business Value	Quantifizierter Wert einer Funktion für das Unternehmen
Changemanagement	Veränderungsmanagement, Planung und Durchführung von Softwareveränderungen
Code	Programmierter Teil einer Software (logische Zeichenfolge)
Daily Scrum	Tägliches Meeting für die Abstimmung und Synchronisation innerhalb des Scrum-Teams
Design	Skizzieren eines Vorhabens/Entwicklungsschrittes
EDI	Electronic Data Interchange, Elektronischer Datenaustausch
Entwicklungszyklus	Zeitabschnitt / Zusammenfassung von Softwareerweiterungen oder Softwareanpassungen
Epic	Zusammenfassung von mehreren Kernfunktionen einer Software
ERP	Enterprise-Resource-Planning
ERP Modul	Bestandteil eines ERP-Systems, meist funktionale Unterteilung
ERP-Consultant	Spezialisierte Unternehmensberater/innen für ERP-Systeme
ERP-Entwicklung	Weiterentwicklung von ERP-Softwarecodes (Kern der Software) durch den ERP-Systemanbieter
ERP-Projekt	Im Kontext dieser Arbeit: Einführungsprojekt für ein ERP-System

Explorative Tests	Erforschende, experimentelle Tests
Fachbereich	Bereich / Abteilung eines Unternehmens
Fachspezialist/in	Spezialisierte Person in einer Abteilung / Unternehmensbereich
Feature	Kernfunktionen einer Software, Zusammenfassung mehrerer gleichartigen Systemfunktionalitäten
Fehlerklassifizierung	Ausprägung / Kategorisierung eines Fehlers
Fertigstellungsgrad	Fortschritt einer oder mehrerer Aufgaben
Funktionalitäten	Zusammenfassung mehrerer Anforderungen / Funktionen
Hotfix	Schnelle / sofortige Reparatur eines Softwarefehlers
Inkrementell	Laufend, zu Bestehendem addierend
Integrate	Zusammenführung von entwickelten Funktionalitäten in einem Gesamtsystem
Integrations-Umgebung	Nicht produktiv eingesetzte Softwareumgebung. Umgebung, wo sämtliche Entwicklungsstände zusammengeführt werden.
IT	Informationstechnik
Komplexität	Schwer vorhersehbares Zusammenspiel von Menschen und Systemen
Konzeptphase	Erstellung eines Lösungsentwurfs
O365	Microsoft Office 365
Performancetest	Testen der Leistungsfähigkeit / Reaktionszeiten von Software
Persona	Rolle / Ausprägung einer fiktiven Person
Product Backlog	Auflistung aller Stories, welche in einem agilen Vorhaben potentiell umgesetzt werden können
Product Owner	Produkteigentümer, Person in Scrum, welche unter Anderem Stories priorisiert
Produktivbetrieb	Betrieb der Software im realen Umfeld
Prozessorientiert	Unternehmensform, welche sich stark nach den Abläufen im Unternehmen richtet
Regressionstest	Wiederholte Durchführung derselben Tests zur permanenten Sicherstellung spezifischer Funktionalitäten
Release (ERP)	Auslieferung von Weiterentwicklungen durch den ERP-Systemanbieter an einen oder mehrere Kunden
Release (SCRUM)	Übergabe von fertig entwickelten Stories und Features an den Produktivbetrieb
Release Backlog	Zeitliche Planung / Auflistung der Übergabe von fertig entwickelten Stories und Features an den Produktivbetrieb

Release Plan	Zeitliche Planung / Auflistung der Übergabe von fertig entwickelten Stories und Features an den Produktiv-Betrieb
Reporting	Berichterstattung / Auswertung des Ist-Zustandes
Requirements Engineering	Erstellung und Pflege von Softwareanforderungen
Retests	Wiederholte Durchführung derselben Tests
Scope	Umfang
Scrum	Vorgehensmodell in der agilen Entwicklung
Scrum Framework	Gerüst / Rahmen von Scrum
Scrum Master	Coach innerhalb eines Scrum-Teams
Shippable Product	Auslieferbare Softwarekomponente
Sicherheitstest	Test im Zusammenhang mit der Sicherheit von Software (IT-Security)
Sprint	Gleichbleibender und sich wiederholender Zeitabschnitt im Scrum-Framework, in welchem vordefinierte Stories umgesetzt werden
Sprint Planning	Planung eines Entwicklungszyklus (Sprint) in Scrum
Stabilität (System)	Verlässlichkeit der Software bezüglich Ausfälle / Störungen / Erreichbarkeit
Stakeholder	Anspruchsberechtigte Person
Stammdaten	Grundinformationen innerhalb von Software
Stories "schneiden"	Anpassung der Grösse (Umsetzungsaufwand) einer Story
Story	Geschichte / grob formulierte Anforderung(en) im Scrum-Framework
Systemkonfiguration	Einstellungen / Parametrisierung von Standardsoftware
Taskboard	Aufgabentafel innerhalb eines Sprints
TDD	Test Driven Development
Test Case	Testfall
Testing	Überbegriff für sämtlicher Testaktivitäten
Testsession	Ausführung von mehreren aneinander folgenden Tests als Einzelperson oder in der Gruppe
User Story	Geschichte / grob formulierte Anforderung(en) im Scrum-Framework
Wasserfall (Modell)	Lineares, definiertes Vorgehen bei der Umsetzung von Projekten
Werkzeug	Für spezifische Aufgaben spezialisierte Software
Workflow	Prozess / digitalisierter teil- oder vollautomatisierter Prozess

1 Ausgangslage und Abgrenzungen

Agilität gewinnt je länger je mehr Bedeutung und ist im beruflichen Alltag nicht mehr wegzudenken. In ERP-Einführungsprojekten ist Agilität jedoch eher Neuland. Es sind einige Berichte zu lesen, in welchen agile Ansätze in ERP-Projekten angewendet wurden, jedoch werden erfahrungsgemäss noch wesentlich mehr ERP-Projekte «klassisch» nach dem Wasserfall-Modell abgewickelt. Aus diesem Grund wurde das Themengebiet «Agilität und ERP» in der Literatur noch wenig behandelt.

Die Einführung eines ERP-Systems hat einen hohen Impact auf ein Unternehmen. Nebst organisatorischen und technischen Aspekten nimmt das Testing einen wichtigen Stellenwert ein. Unternehmen, welche sich für eine agile Umsetzung eines solchen Projektes entscheiden, betreten oft Neuland. In dieser Masterarbeit wird der Bereich Testing in agilen ERP-Projekten mit Hilfe von Fachliteratur sowie Experten genauer analysiert. Als Resultat wird eine praxistaugliche Testkonzeption für mittelgrosse ERP-Projekte entwickelt. Mit der Verknüpfung dieser Fachgebiete kann für Unternehmen, welche sich für den agilen Ansatz in einem ERP-Projekt entscheiden, erheblichen Mehrwert und Sicherheit geschaffen werden.

Folgende Fragestellungen sind zentral:

- Welchen Einfluss hat agiles Vorgehen in einem ERP-Projekt auf das Testing?
- Welche organisatorischen Voraussetzungen müssen in einem Unternehmen für ein erfolgreiches Testing im einem agilen ERP-Projekt erfüllt werden?
- Wie wird das Testmanagement konzipiert und welche Methoden und Techniken eigenen sich in einem agilen ERP-Projekt?
- Wie kann die Qualität der Software aus integrativer Sicht mit produktiv laufenden Umsystemen sichergestellt werden?
- Wie werden Testfälle und Ergebnisse dokumentiert damit sie nach Projektabschluss in den operativen Betrieb überführt werden können?

Aufgrund der Beantwortung der oben erwähnten Fragestellungen werden mit dieser Arbeit folgende Zielsetzungen erreicht:

- Es wird ein mögliches Vorgehen für die Erstellung einer Testkonzeption in einem agilen ERP-Projekt beschrieben.
- Es wird aufgezeigt, wie Testing in einer agilen ERP-Projektorganisation eingebettet werden kann.
- Es sind geeignete Massnahmen bekannt, wie die Qualität der Integration von produktiv laufenden Umsystemen sichergestellt wird.

1.1 Vorgehen

Die Literaturrecherche hat gezeigt, dass zu den Themengebieten Agilität und Testing zwar Literatur vorhanden ist, jedoch nicht spezifisch zum Thema agiles Testen in ERP Projekten. Um diese Lücke zu füllen, und zusätzlich dem Anspruch einer wissenschaftlichen Arbeit gerecht zu werden, wurden unterschiedliche Experten in Form von Interviews befragt. Die Experten wurden so ausgewählt, dass das zu erforschende Themengebiet aus unterschiedlichen Blickwinkeln betrachtet werden kann. Die Ergebnisse aus diesen Interviews wurden an unterschiedlichen Stellen, sowohl im theoretischen Teil als auch in den Handlungsempfehlungen zitiert. Sämtliche Interviews sind im Anhang beigefügt.

ERP-Consultant (Daniel Frei, acreo consulting ag)

Daniel weist mehrere Jahre Erfahrung als Projektleiter im ERP-Umfeld aus. Als unabhängiger ERP-Berater begleitet er Unternehmen bei der Evaluation, Beschaffung und Einführung von ERP-Systemen. Er ist Partner und Senior Project Manager bei acreo consulting ag (Acreo Consulting, 2020).

Agile Coach (Klaus Bucka-Lassen, aragost ag)

Klaus ist Certified Scrum Master, Certified Scrum Professional und Certified Scrum Product Owner. Mit seinem im Jahr 2001 gegründeten Unternehmen begleitet er diverse Unternehmen bei der agilen und digitalen Transformation (Aragost, 2020).

ERP-Projektleiter (Reto Hansmann, Rhätische Bahn AG)

Reto ist IT-Projektleiter bei der Rhätischen Bahn AG. Er hat in der Vergangenheit bei unterschiedlichen Unternehmen, in der Rolle als Projektleiter, ERP-Systeme eingeführt und weiterentwickelt.

ERP-Systemanbieter (anonymisiert, nachfolgend Hans Meier genannt)

"Hans Meier" wurde in dieser Arbeit auf eigenen Wunsch anonymisiert. Er ist CEO eines ERP-Systemanbieters in der DACH-Region und hat gemeinsam mit seinen Kunden Erfahrung in der Umsetzung von agilen ERP-Projekten gesammelt.

In Kapitel 2 (Testen in agilen Projekten) wurden die theoretischen Grundlagen im Bereich Agilität und Testing erarbeitet. Diese beziehen sich hauptsächlich auf die Literatur sowie die durchgeführten Interviews.

In Kapitel 3 (ERP-Projekte) wurde der Bezug zum Themengebiet ERP hergestellt. Aufgrund der eigenen mehrjährigen Erfahrung als Fachspezialist ERP sowie der Interviewpartner wurde auf die spezifischen Eigenheiten und Merkmale von ERP-Projekten und ERP-Systemen eingegangen.

In den Handlungsempfehlungen (Kapitel 4) wurden die beiden Themengebiete vereint und konkrete Vorschläge für agiles Testing in ERP-Projekten ausgearbeitet.

Mit dem abschliessenden Fazit (Kapitel 4) wurde das eigene Vorgehen kritisch reflektiert und die ausgearbeiteten Ergebnisse bewertet.

1.2 Abgrenzungen

Da für diese Arbeit ein agiles Referenz-ERP-Projekt gewählt wurde, wird in dieser Arbeit hauptsächlich auf das Scrum-Framework verwiesen, da sich das Referenz-Unternehmen bereits für dieses Framework entschieden hat und dieses in agilen Unternehmen tendenziell präferiert angewendet wird (Meier, 2020). Somit werden keine weiteren agile Methoden untersucht und beschrieben.

Die Evaluation von Werkzeugen für Testautomatisierung und Testdokumentationen steht nicht im Fokus dieser Arbeit. Es werden zwar Empfehlungen für die Anwendung von Werkzeugen abgegeben, technologische Unterschiede spezifischer Werkzeuge sowie deren Vor- und Nachteile werden nicht untersucht.

Die Beurteilung und Gegenüberstellung von IT-spezifischen Technologien, im speziellen Schnittstellen-Technologien, waren nicht Teil des CAS-Lehrganges und werden entsprechend nicht behandelt. Im Fokus dieser Arbeit stehen organisatorische- und methodische Aspekte im Kontext zum gewählten Themengebiet.

Obwohl Branchenunterschiede einen starken Einfluss auf die Wahl des ERP-Systems sowie auf dessen Prozesse ausüben, werden diese nicht detailliert behandelt. Es wird angenommen, dass die erstellten Handlungsempfehlungen in agilen ERP-Projekten branchenübergreifend angewendet werden können. Ausgenommen sind explizit hoch regulierte Branchen, welche hohe und spezifische Anforderungen an Testverfahren voraussetzten.

1.3 Referenz ERP-Projekt

Agile ERP-Projekte unterscheiden sich im Wesentlichen bezüglich ihres Projektumfangs. Je grösser der Umfang, desto länger dauert das Projekt respektive desto mehr agile Teams arbeiten gleichzeitig an demselben Projekt. Es gäbe natürlich noch weitere Unterscheidungsmerkmale wie beispielsweise Branchenunterschiede. Jedoch sind diese im Rahmen des Testing nicht weiter relevant. Einzig hohe gesetzliche Anforderungen wie beispielsweise bei Medtech-Unternehmen oder bei Banken könnte einen zusätzlichen Einfluss haben, insbesondere auf die Dauer, die Dokumentationspflicht und den Umfang der Tests.

In dieser Arbeit wird als Referenz ein agiles ERP-Projekt eines Unternehmens im öffentlichen Verkehr verwendet. Ziel dieses Vorhabens ist die Ablösung von mehreren bestehenden ERP-Lösungen und Applikationen. Mit diesem Vorhaben möchte das Unternehmen den Wertefluss standardisieren und gleichzeitig eine solide Grundlage für die zukünftigen Herausforderungen im digitalen Zeitalter schaffen.

Mit rund 1700 Mitarbeiter und einem breiten Tätigkeitsfeld ist dieses Projekt als mittelgross bis gross einzustufen.

Abbildung 1: Projektumfang ERP-Projekt.

Quelle: Eigene Darstellung.

Aufgrund des hohen funktionalen Umfangs und relativ kurzem Umsetzungszeitraum (ca. zwei Jahre) wird das Projekt mutmasslich mit mehreren Scrum-Teams abgewickelt. Die Projektorganisation für die Auswahl des Systemanbieters ist bereits gesetzt. Für die Umsetzungsphase wurden noch keine Scrum-Teams gebildet. Diese Arbeit soll aufzeigen, wie die Test-Organisation bei agilen ERP-Projekten mit ähnlichem Umfang aufgestellt werden kann.

2 Testen in agilen Projekten

In diesem Kapitel werden die theoretischen Grundlagen zu Agilität und Testing erarbeitet. Nebst der Darlegung von Unterschieden zwischen dem Wasserfall-Modell und agilem Vorgehen werden die einzelnen Elemente des Scrum-Frameworks beschrieben. Weiter wird in diesem Kapitel aufgezeigt, welche Methoden und Techniken angewendet werden können und welche Dokumente im Verlaufe der Test erstellt werden sollten. Am Ende dieses Kapitels wird kurz beschrieben, welche vorbereitenden Tätigkeiten im Rahmen der agilen Tests durchgeführt werden sollten.

2.1 Grundlagen Agilität

Bevor spezifisch auf das Themengebiet «Agile Testing» eingegangen wird, sollten als erstes einige Grundprinzipen der Agilität aufgezeigt werden. Immer mehr Unternehmen verfolgen heute agile Ansätze bei der Umsetzung ihrer Vorhaben. Vor Allem in Bereichen wo Komplexität zunimmt, steigt die Notwendigkeit für agile Ansätze (Bucka-Lassen, 2020). Heuer (2014) beschreibt jedoch, dass Unkenntnis bezüglich der agilen Arbeitsweise oft zum Misserfolg und Problemen führen kann (S. 7). «Agile isn’t something you do. It’s something you are. » (Heuer, 2014, S. 5). So ist Agilität, respektive das Scrum-Framework, ein Ansatz, der einen starken Einfluss auf die Organisation ausübt und die Menschen sowie derer Grundhaltung und Einstellung wesentlich verändert. (Gloger, 2016, S. 19). Die Beweggründe für die Entstehung der agilen Bewegung werden im nachfolgenden Absatz kurz aufgezeigt.

2.1.1 Vom Wasserfall zur Agilität

In den späten 70er und 80er Jahren wurde Software meist mit dem Wasserfall-Modell entwickelt. Auch heute wenden noch viele Unternehmen dieses Modell an. Dies auch darum, weil sich dieses Modell über die Jahre erfolgreich etabliert und früher prinzipiell keine anderen vernünftigen Methoden bekannt waren (Heuer, 2014, S. 7). Wer jetzt befürchtet, dass eine lange Abhandlung zum Wasserfall-Modell verfasst wurde sei beruhigt. Das Wasserfall-Modell dient nur als Überleitung zur Agilität, im speziellen zum agile Testing und der Darlegung der Gründe, wieso der agile Ansatz entstanden ist.

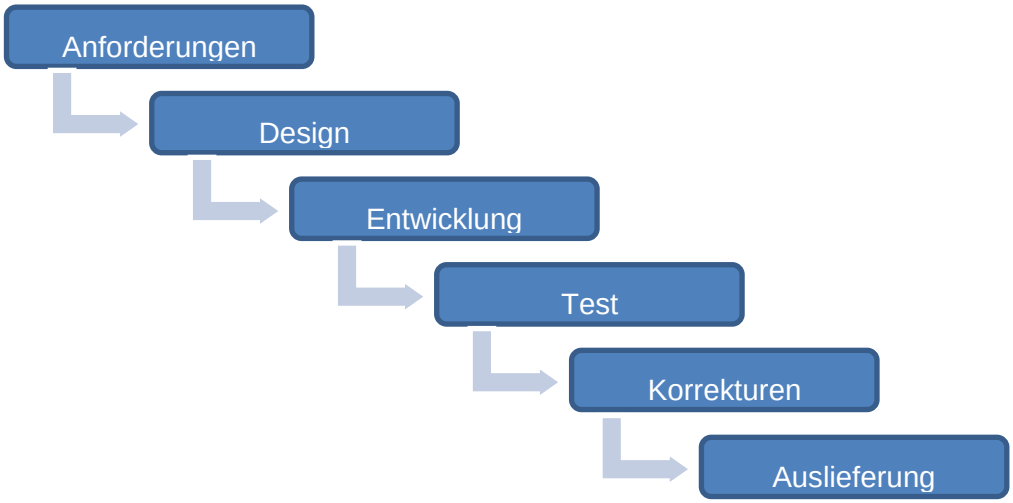

Abbildung 2: Wasserfall-Modell in der Software-Entwicklung.

Quelle: In Anlehnung an Heuer (2014, S. 7)

Obwohl das Wasserfall-Modell über Jahrzehnte erfolgreich angewendet wurde und auch heute noch in Gebrauch ist, hat es doch wesentliche Schwachpunkte. Einer davon ist beispielsweise, dass Tester/innen und Entwickler/innen nicht in demselben Team respektive nicht in derselben Projektphase arbeiten. Somit wird das Projekt nach Identifikation von Fehlern mehrfach der Entwicklung übergeben, um es anschliessend wieder durch die Testorganisation prüfen zu lassen. Diese Tatsache gilt als die Hauptursache von Projektverzögerungen in entsprechenden Projekten (Heuer, 2014, S. 7). Des Weiteren ist es in solchen Projekten schwierig grundlegende Fehler in der Konzeptphase nachträglich zu korrigieren (Heuer, 2014, S. 8). Der aus Sicht des Schreibenden grösste Nachteil ist jedoch die Dauer, mit welcher das Produkt an den Markt gebracht werden kann. In Wasserfall Projekten findet keine inkrementelle Lieferung statt. So wird das Produkt erst ausgeliefert, wenn es komplett erstellt wurde. Ein Unternehmen kann es sich heute nicht mehr leisten, ein Produkt erst nach langer Entwicklungszeit zu verkaufen. Dazu sind Mitbewerber zu flexibel und die Technologien und Bedürfnisse am Markt verändern sich schneller, als dass es nach Wasserfall umgesetzt werden kann, was auch (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos (2018) bestätigen (S. 16). Bucka-Lassen (2020) erwähnt zudem, dass beispielsweise zu Beginn einer Softwareeinführung das Wissen für den Endzustand der Software noch wenig ausgeprägt ist. Entsprechend können wichtige Entscheidungen in einem frühen Zeitpunkt, wie es nach dem Wasserfall-Modell üblich ist, nicht kompetent getroffen werden.

Um sich von den strengen Einschränkungen des Wasserfalls zu befreien, und die bekannten Schwierigkeiten zu eliminieren, entwickelten Fachexperten im Jahr 2001 das «Agile Manifest». Inkrementelle Lieferung, flexible Arbeitsteilung sowie Verantwortung für das Entwicklungs-Team als Ganzes waren einige der Grundprinzipien (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018).

Gloger (2016) beschreibt in seinem Buch folgende Werte für die agile Software Entwicklung, abgeleitet von http://agilemanifesto.org (S. 38):

- **«Individuen und Interaktion** stehen über Prozessen und Werkzeugen»
- **«Funktionierende Software** steht über umfangreicher Dokumentation»
- **«Die Zusammenarbeit mit dem Kunden** steht über der Verhandlung von Verträgen»
- **«Das reagieren auf Veränderung** steht über dem Befolgen eines Plans»

Die Revolutionäre Idee dahinter war, dass Requirements-Engineering, Entwicklung und Testing nicht mehr durch unterschiedliche Personen in unterschiedlichen Phasen umgesetzt wurden, sondern durch dasselbe Team, idealerweise in demselben Raum. Sinnbildlich dafür ist der aus dem Rugby stammende Begriff Scrum, welcher symbolisch dafürsteht, mit Teamzusammenhalt gegenüber dem Gegner Raum zu gewinnen (Gloger, 2016, S. 24).

Abbildung 3: Scrum - Stormers gegen Bulls, Cape Town 16.02.2008.

Quelle: Gloger (2016, S. 24)

2.1.2 Scrum-Framework

Im Laufe der Zeit hat sich das Scrum-Framework als bevorzugte agile Methode in vielen Unternehmen etabliert (Frei, 2020). Die Abbildung 4 zeigt, wie Scrum grundsätzlich aufgebaut ist und welches die grundlegenden Komponenten sind.

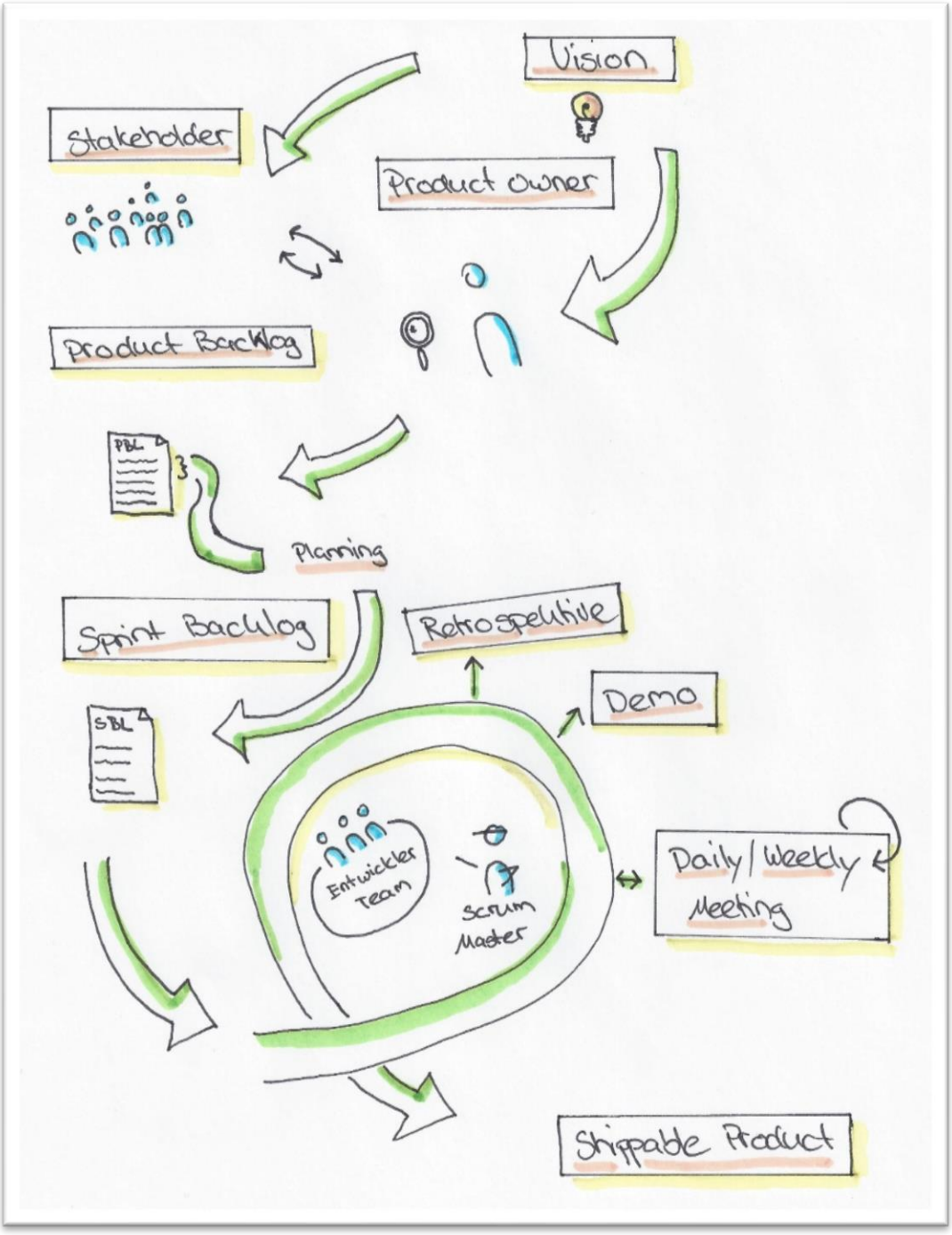

Abbildung 4: Scrum Framework.

Quelle: In Anlehnung an Heuer (2014, S. 11)

Da in dieser Arbeit der Fokus auf dem Testing liegt, dienen die nachfolgenden Ausführungen lediglich der Erklärung einiger Begrifflichkeiten, auf welche in den nachfolgenden Kapiteln jeweils Bezug genommen wird. Wichtig ist hier zu erwähnen, dass Scrum

lediglich ein Gerüst respektive ein Rahmenwerk darstellt. Das effektive Vorgehen innerhalb dieses Rahmens definiert das Unternehmen respektive das Team selbst (Bucka-Lassen, 2020).

2.1.2.1 Vision

Die Vision ist ein zentrales und sehr wichtiges Element von Scrum. Sie soll die Beteiligten Menschen begeistern, damit alle in dieselbe Richtung marschieren. Ein Beispiel für eine Vision, welche Menschen bewegen kann, hat beispielsweise Martin Luther King mit seiner bekannten Rede mit dem Titel «I Have a dream» eindrücklich unter Beweis gestellt. Diese Vision hatte massgeblich zur Aufhebung der Rassentrennung beigetragen (Wikipedia, 2019). Auch die Mission Mondlandung der Amerikaner startete mit einer starken Vision von John F. Kennedy: «I believe that this nation should commit itself to achieving the goal, before this decade is out, of landing a man on the Moon and returning him safely to Earth. (Gloger, 2016, S. 143)»

Auch wenn die Mondlandung hier ein extremes Beispiel ist, sollte der Product Owner sich vertieft mit der Vision auseinandersetzten und die Werte und Vorstellungen des Produktes in starke Worte fassen (Gloger, 2016, S. 144). So kann er die Mitarbeiter vereinen und möglicherweise ein grossartiges Produkt erschaffen.

2.1.2.2 Product Owner und Stakeholder

Product Owner und Stakeholder sind in Scrum stark miteinander verbunden. Product Owner sind verantwortlich, dass das Produkt nach den Vorstellungen der Stakeholder entwickelt wird und nehmen somit eine Schlüsselposition zwischen Kunden und Entwickler/innen wahr. In engem Austausch mit den Stakeholdern sind Product Owner dafür verantwortlich, den Product Backlog zu führen (Gloger, 2016, S. 100). Zusätzlich stehen sie in der Sprint-Planung in engem Austausch mit dem Entwicklungs-Team. So haben Product Owner nebst analytischen auch viele kommunikative Aufgaben zu bewältigen.

2.1.2.3 Product Backlog

Der Product Backlog ist im weiteren Sinne eine Konkretisierung der Vision. Darin sind die Eigenschaften und Funktionalitäten des Produktes enthalten (Gloger, 2016, S. 103). Wichtig ist hierbei, dass es sich nicht um klassisches Requirements Engineering handelt, wo bereits sehr detailliert die Funktionalitäten beschrieben sind. Viel mehr sind im Backlog so genannte «Geschichten» respektive User Stories verfasst, welche keine Designvorschriften oder Ähnliches enthalten. Diese werden erst zu einem späteren Zeitpunkt durch das Entwicklungs-Team ausgearbeitet (Gloger, 2016, S. 104). In einem zweiten Schritt werden die User Stories ihrem Wert (Business Value) priorisiert. Im Normalfall geschieht die Priorisierung zwischen den Stakeholdern und dem Product Owner. Bei grösseren Projekten kann auch ein Steuerungsgremium zum Zwecke der Priorisierung eingesetzt werden (Ritterkamp, Schmitz-Ohrndorf, Arndt, Harstick, & Knapp, 2013, S. 7). So wird durch den Product Owner sichergestellt, dass das Entwicklungs-Team die

aus Sicht der Organisation wertvollsten Stories als erstes umgesetzt werden (Gloger, 2016, S. 100).

2.1.2.4 User Stories

Wie bereits erwähnt, stehen ganz zu Beginn die User Stories. Diese werden durch die Stakeholder in natürlicher Sprache verfasst und sind an wenige Vorgaben gebunden. Im Normalfall ist eine User Story in einem Satz formuliert könnte wie in folgendem Beispiel aussehen:

«Als ein Einkäufer möchte ich eine Bestellung in Form eines PDF-Dokumentes erzeugen»

Trotz der Einfachheit der Formulierung sollten unterschiedliche Punkte beachtet werden. Als erstes muss darauf geachtet werden, dass die Stories nicht zu gross sind. Dieser Punkt ist wesentlich, und hat Auswirkungen auf das gesamte Testing. Wenn die Story zu gross wird (1-2 Wochen Entwicklung) besteht die Gefahr, dass die Story in einem Sprint nicht fertig gestellt werden kann. Zudem werden die Testaktivitäten je Story grösser und benötigen entsprechend mehr Zeit. Eine Story ist erst fertig, wenn sie getestet wurde und entsprechend die Akzeptanzkriterien erfüllt sind. Ansonsten muss sie wieder dem Backlog übergeben werden. Die Abbildung 6 zeigt die Auswirkung von grossen Stories im Vergleich zu kleinen. Wo bei der ersten Grafik nur eine Story (ca. 40%) fertig gestellt wurde, liegt bei der zweiten Grafik der Fertigstellungsgrad bei 5 Stories (ca. 90%). (Gregory & Crispin, 2015, S. 310)

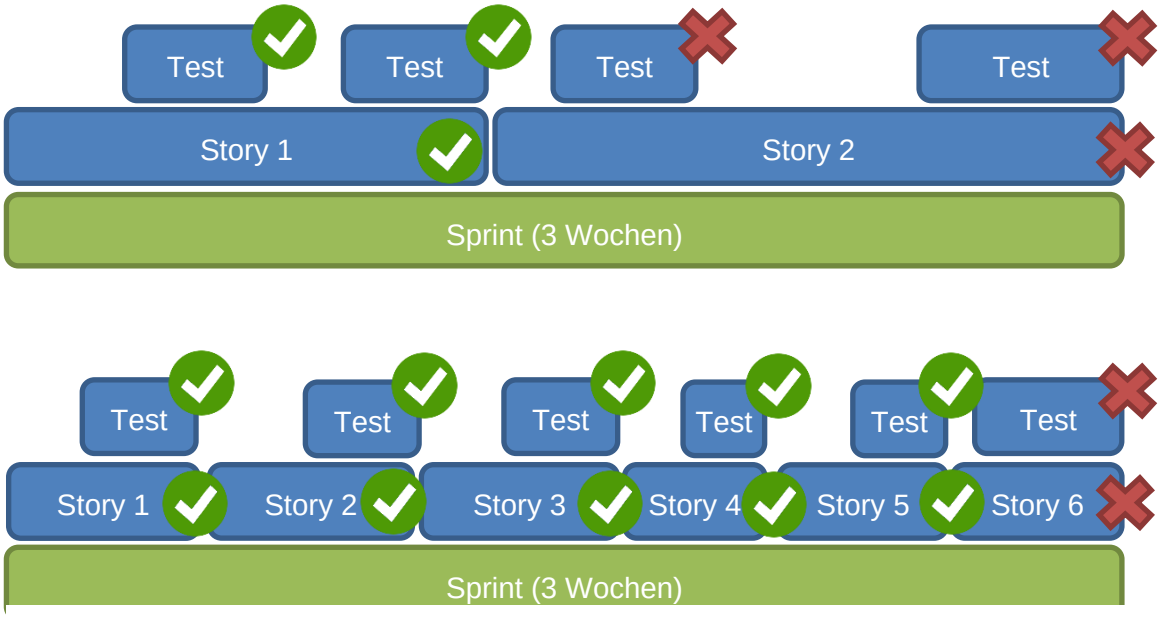

Abbildung 5: Auswirkung unterschiedliche Grösse von Stories.

Quelle: Eigene Darstellung

Die Skalierung der Stories kann bereits zu Beginn durch den Product Owner wahrgenommen werden. Heuer (2014) beschreibt, dass dies auch eine gute Gelegenheit für die Tester/innen ist, mit den Stakeholder erstmalig ins Gespräch zu kommen und die effektiven Bedürfnisse bezüglich dieser Stories zu erfahren. Denn mit dem Fokus auf das Testing kann relativ gut abgeschätzt werden, wie gross die Stories effektiv sind (S. 25). Spätestens im Sprint Planning ist die Gelegenheit, grosse Stories in kleinere zu unterteilen, damit diese durch das Team in einem Sprint ungesetzt werden können. Dazu werden jeder Story sogenannte Story Points zugewiesen. Diese zeigen den Umsetzungsaufwand in Relation zu den anderen Stories (Ritterkamp, Schmitz-Ohrndorf, Arndt, Harstick, & Knapp, 2013, S. 8).

Um die Story auf ihre Qualität zu beurteilen, kann das Akronym INVEST angewendet werden:

I-Independent	Jede User Story muss für sich stehen. Dies erleichtert die Zuweisung von Aufgaben zu Sprints.
N-Negotiable	Die Story ist nicht in Stein gemeisselt. Es steht jedem frei, Änderungen zu besprechen.
V-Valuable	Die User Story muss einen Mehrwert bringen.
E-Estimable	Es soll möglich sein, für diese Story eine grobe Aufwandschätzung zu erstellen.
S-Small	Die Story muss klein genug sein, um daraus konkrete Aufgaben abzuleiten
T-Testable	User Stories müssen so formuliert sein, dass daraus klare Anforderungen und Testfälle abgeleitet werden können.

Tabelle 1: Akronym INVEST.

Quelle: In Anlehnung an Heuer (2014, S. 25)

Ein weiterer Ansatz, um qualitative Stories zu verfassen ist der Einsatz von **«Power of Three»**. Gregory & Crispin (2015) erwähnen, dass eine Story durch unterschiedliche Blickwinkel betrachtet werden kann; Durch den Entwickler, den Tester und den Product Owner respektive Stakeholder (S. 368). Es gilt als ein sehr gutes Zeichen, wenn sich diese drei Personen betreffend Inhalt der Story einig sind. Auch zu einem späteren Zeitpunkt, im Sprint Planning oder in den Sprints, ist der Austausch zwischen diesen drei Rollen ein wertvolles Werkzeug, wenn Probleme oder Unstimmigkeiten auftauchen (Gregory & Crispin, 2015, S. 369).

2.1.2.5 Sprint Planning und Sprint Backlog

Bevor mit dem eigentlichen Sprint begonnen wird, finden das genannte Sprint Planning statt. Sobald der Product Backlog definiert und priorisiert ist, werden die User Stories mit dem Entwicklungs-Team besprochen. Im Grundsatz werden die Prioritäten aufgrund des Product Backlog abgearbeitet. Jedoch hat das Team die Möglichkeit, im Rahmen des Sprint Planning die Priorisierung teilweise zu beeinflussen (Ritterkamp, Schmitz-Ohrndorf, Arndt, Harstick, & Knapp, 2013, S. 7). In einem ersten Meeting ist das gesamte Team inklusive Product Owner, Anwendervertreter und Fachverantwortliche anwesend (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 147). Gemeinsam mit den Anwesenden wird skizziert, wie die Stories umgesetzt werden sollen. Schritt für Schritt werden mehr Informationen gesammelt, damit deren Umsetzungs-Aufwand besser geschätzt werden kann (Gloger, 2016, S. 175).

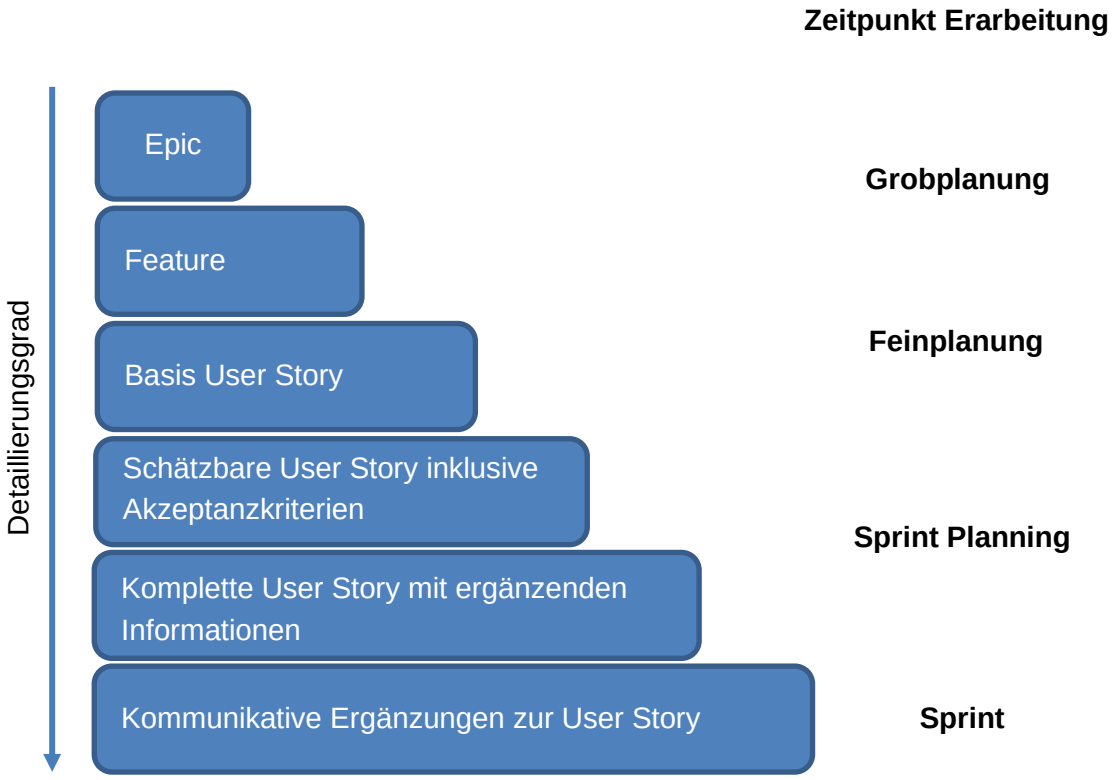

Abbildung 6: Detaillierung von Stories und Anforderungen

Quelle: In Anlehnung an Ritterkamp, Schmitz-Ohrndorf, Arndt, Harstick, & Knapp (2013, S. 11)

Aufgrund des geschätzten Aufwandes, sowie der Kapazität des Entwickler Teams kann im Sprint Backlog (auch Selected Product Backlog) anschliessend fixiert werden, welche Stories im nächsten Sprint umgesetzt werden (Gloger, 2016, S. 188). Aufgrund dieser Informationen kann zusätzlich der Releaseplan aktualisiert werden. In diesem ist ersichtlich wann den Stakeholdern welche Funktionalitäten zur Verfügung stehen. Hier ist zu beachten, dass sich die Prioritäten im Verlaufe der Sprints verändern können. Je näher der Sprint, desto genauer sind die Release-Informationen.

Die Teammitglieder beschreiben in geeigneter Form und Detaillierungsgrad die Stories. Zusätzlich werden in diesem Schritt die Akzeptanzkriterien definiert (Gloger, 2016, S. 190). Hier ist es empfehlenswert, die Detaillierung der Akzeptanzkriterien und den dazugehörigen Tests zu diesem Zeitpunkt zu machen, bevor an den Stories gearbeitet wird. Diese Informationen definieren zu einem wichtigen Teil die Anforderungen bezüglich der Story (Robson, 2016, S. 1372).

An diesem Punkt unterscheiden sich grosse von kleinen Stories wesentlich. Wenn die Stories genug klein «geschnitten» sind, sind Akzeptanzkriterien für die Abnahme der Stories ausreichend (siehe Abbildung 8).

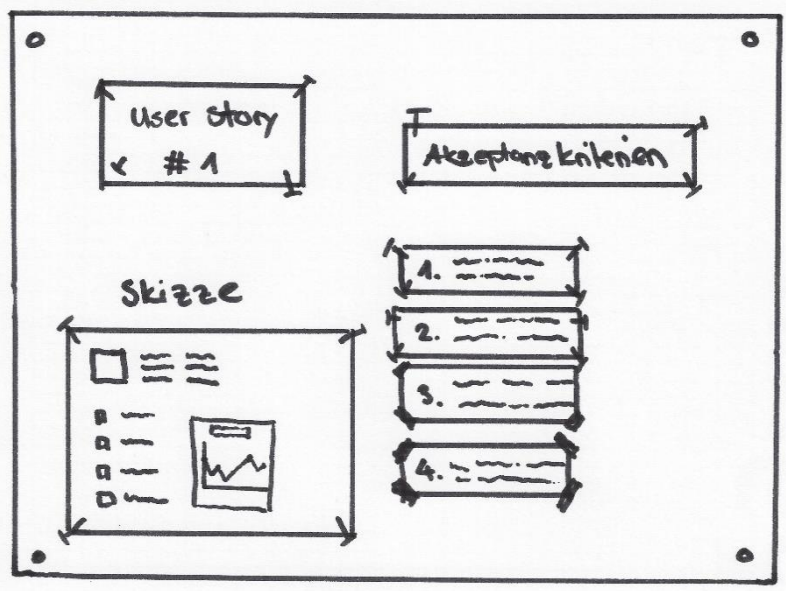

Abbildung 7: Anforderungen und Test für kleine User Stories

Quelle: In Anlehnung an Gloger (2016, S. 188)

Grosse Stories hingegen bedingen mehrere und kompliziertere Testfälle und auch Anforderungen müssen detailliert beschrieben werden (siehe Abbildung 8).

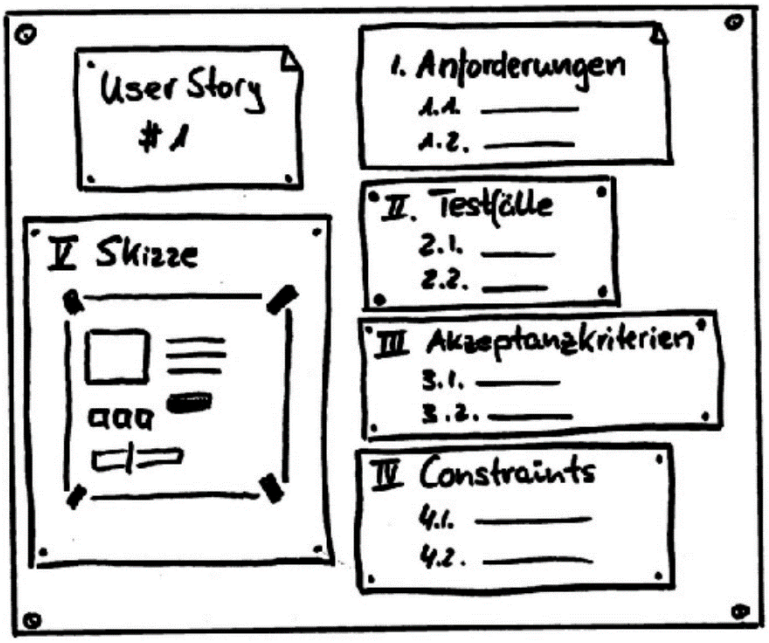

Abbildung 8: Anforderungen und Test für grosse User Stories
Quelle: Gloger (2016, S. 188)

Tendenziell werden die Stories eher zu gross als zu klein zu «geschnitten». Dabei besteht die Gefahr, innerhalb der Projektes «kleine Wasserfälle» zu kreieren und einige Stories über mehrere Sprints hinweg zu bearbeiten. Aus diesem Grund sollte spätestens im Sprint Planning grosse Stories in kleinere unterteilt werden.

2.1.2.6 Definition of Done und Akzeptanzkriterien

Die Definition of Done ist ein Statement des Teams, dass die entwickelte Story fertig ist. Sie wird grundsätzlich für alle User Stories gleichermassen definiert und ändert sich im Verlaufe der Sprints nur wenig. Gloger (2016) erwähnt, dass die Definition of Done im Idealfall durch das Scrum-Team festgelegt wird (S. 206). Beispiele für die Definition of Done sind:

- Der entwickelte Code ist im System XY dokumentiert
- Die Akzeptanzkriterien sind erfüllt
- Regressionstests wurden aus den Testfällen abgeleitet und sind dokumentiert
- Die Entwicklungen sind in der Integrations-Umgebung eingespielt

Im Gegensatz zu der Definition of Done sind die Akzeptanzkriterien je Story unterschiedlich. In diesen werden konkrete Lösungseigenschaften gefordert, welche durch die Story erfüllt werden müssen. Diese Lösungseigenschaften müssen so formuliert werden, dass sie mit Tests überprüft werden können (Ritterkamp, Schmitz-Ohrndorf, Arndt, Harstick, & Knapp, 2013, S. 8). Baumgartner (2018) erwähnt, dass auch Antwortzeiten und

Stabilität ebenfalls als Akzeptanzkriterien vermerkt werden können (S. 115). Beispiele für Akzeptanzkriterien sind:

- Die Bestellung kann als PDF-Dokument erzeugt werden
- Die Rechnungsadresse des Kunden wird in der Faktura korrekt angezeigt
- Die Antwortzeit des Webshops für die Erstellung der Rechnung darf 5 Sekunden nicht überschreiten

2.1.2.7 Sprint

Im eigentlichen Sprint werden die im Planning definierten Stories und Aufgaben fokussiert umgesetzt. Im Normalfall dauert ein Sprint 2-4 Wochen. Dies kann je nach Organisation und Komplexität unterschiedlich sein.

Die Umsetzung erfolgt im Idealfall mit Hilfe von Karteikarten auf einem Taskboard, auf welchem die Aufgaben sowie dessen aktueller Stand ersichtlich sind.

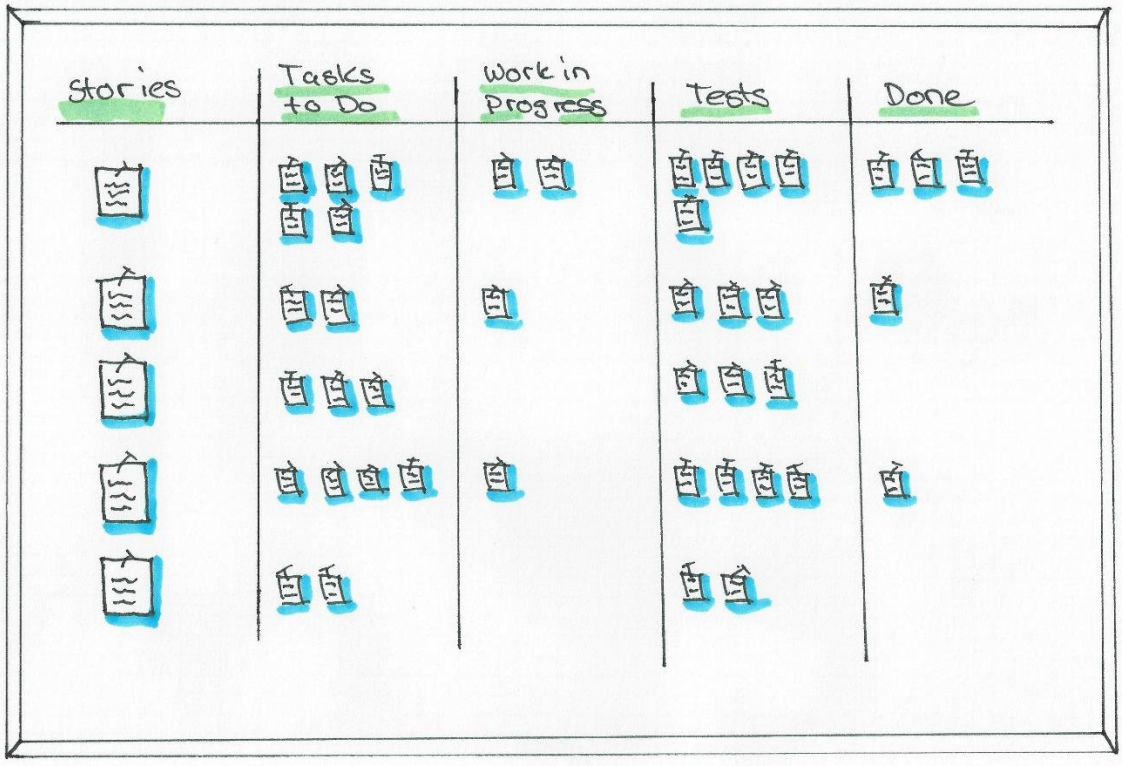

Abbildung 9: Taskboard

Quelle: In Anlehnung an Gloger (2016, S. 193)

An diesem Taskboard ist wichtig, dass die zu erledigenden Aufgaben und Tests klar einer Story zugewiesen sind. Die Stories sollten aufgrund ihrer Priorisierung von oben nach unten gelistet werden. Rechts zu den Stories sind die dazugehörigen Aufgaben und Testfälle notiert (Gloger, 2016, S. 193). Im Verlaufe des Sprints wandern die Karteikarten jeweils von links nach rechts. Sofern die Aufwandschätzungen korrekt waren, sind am Ende des Sprints sämtliche Karteikarten in der Spalte «Done».

2.1.2.8 Entwicklungs-Team

Das Entwicklungs-Team ist das zentrale Element von Scrum. Im Gegensatz zum Taylorismus organisiert sich das das Entwicklungs-Team völlig autonom und selbständig, mit hoher Eigenverantwortung. Gloger (2016) erwähnt, dass das Entwicklungs-Team das zu liefern hat, was sie sich im Sprint vorgenommen haben. Dieses «Commitment» sei eine Selbstverpflichtung jedes einzelnen Teammitgliedes (S. 87). Auch die Fähigkeiten der Teammitglieder sollten möglichst unterschiedlich und umfassend sein (Frei, 2020). Es ist nicht zielführend, wenn das gesamte Team aus Datenbank-Spezialisten besteht, und gleichzeitig die Benutzeroberfläche der Applikation gestaltet werden muss. Im Idealfall ist die Summe aller Teammitglieder im Stande, das Produkt komplett zu erstellen und zu testen (Bucka-Lassen, 2020). Entsprechend ist hier auch der Tester/innen Bestandteil des Teams. Zudem ist das Entwicklungs-Team beständig und arbeitet im Normalfall über einen längeren Zeitraum und über mehrere Sprints zusammen. Jedoch kann es durchaus vorkommen, dass im Verlaufe der Sprints vereinzelte Mitglieder hinzukommen oder wegfallen, je nach Anforderungen und Ressourcenbedarf (Gloger, 2016, S. 96).

2.1.2.9 Scrum Master

Der Scrum Master ist für die Effizient des Entwicklungs-Teams verantwortlich. Es schafft für das Team Hindernisse aus dem Weg, eliminiert aktiv Störungen und Blockaden und vermittelt zwischen Product Owner und Entwicklungs-Team (Gloger, 2016, S. 110). Vielleicht ist ein Scrum Master vergleichbar mit einem Coach, der jedoch keine direkte Weisungsbefugnis gegenüber seinen Teammitglieder besitz. Er unterstütz das Team in taktischen Belangen und ist dafür besorgt, dass so wenig externe Einflüsse wie möglich die Teammitglieder beschäftigen. Das Team soll fokussiert an dem Produkt arbeiten, der Scrum Master begleitet das Team prozessual. Gloger (2016) beschreibt den Scrum Master als «den besten Freund des Teams» (S. 110).

2.1.2.10 Shippable Product

Am Ende jedes Sprints sollte ein funktionsfähiges Produkt verfügbar sein. Dies wird als «Potential Shippable Product Increment» bezeichnet (Gloger, 2016, S. 29). Das heisst, zu diesem Zeitpunkt müssen die im Sprint vorgenommenen Stories umgesetzt sowie getestet sein. Ob das Produkt wirklich ausgeliefert wird, hängt von unterschiedlichen Faktoren ab. Nach den ersten Sprints ist möglicherweise noch nicht genügend Funktionalität vorhanden, dass das Produkt den Kunden ausgeliefert werden kann. Zusätzlich spielen möglicherweise auch Marketing-Überlegungen eine wichtige Rolle; Zu welchem Zeitpunkt und bei welchen Kunden das Unternehmen das Produkt lancieren möchte. In der Agilen Welt spricht man hier vom Minimum viable Product (MVP). So bezeichnet man ein Produkt, dessen Kernfunktionalitäten einem bestimmten Kreis an Kunden ausliefern könnte (Gregory & Crispin, 2015, S. 822). Die effektive Auslieferung an die Kunden wird als sogenannter Release bezeichnet.

2.1.3 Feature / Release inkl. Tests

Obwohl in Scrum nicht so vorgesehen, ist es wichtig auf das Thema Feature und Release inclusive deren Tests einzugehen. In der Abbildung 10 ist ersichtlich, dass aus dem Product Backlog als erstes ein Release Backlog erstellt wird. In diesem werden Stories in unterschiedliche Releases (evtl. auch Features) unterteilt. Dies dienen einerseits der Priorisierung und Terminierung der Stories, andererseits können so Story-übergreifende Integrationstests durchgeführt werden. Konkret werden die Stories in den Sprints zwar getestet (Akzeptanzkriterien), jedoch erfolgt in einem separaten Team zusätzliche Integrations- und Benutzerabnahmetests, nachfolgend Feature-Tests genannt. Vor allem wenn mehrere Teams gleichzeitig entwickeln, und das Projekt entsprechend gross ist, sind solche Tests vermutlich unumgänglich.

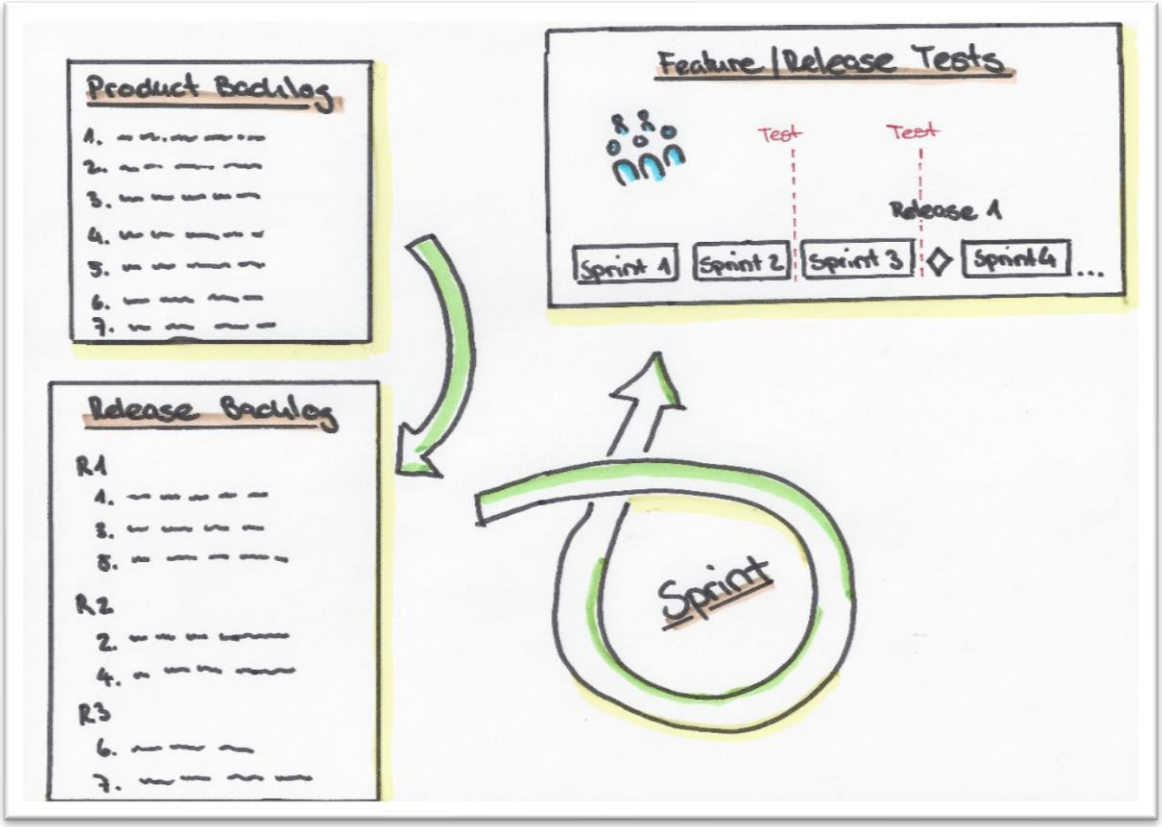

Abbildung 10: Vom Product Backlog zum Feature-Tests

Quelle: In Anlehnung an SwissQ Consulting AG (2020)

2.1.4 Einfluss Agilität auf Testing

Agilität beeinflusst das Testing insofern, dass Entwicklung und Tests nicht mehr als separate Tasks in separaten Prozessen erfolgen, sondern grundsätzlich in demselben Schritt und in demselben Team. Die Vorteile liegen auf der Hand. Fehler werden früh entdeckt und können behoben werden, bevor die Entwickler zu lange «in die falsche Richtung» entwickeln. Durch die enge Zusammenarbeit und Kommunikation ist zudem die Identifikation der Fehler um einiges einfacher und zielführender (Heuer, 2014, S. 8). Daraus resultiert, dass die im Wasserfall-Modell identifizierten langen Test- und Korrekturzyklen massiv verkürzt werden und entsprechend auch das Produkt schneller auf den Markt gebracht werden kann, was natürlich auch durch die inkrementelle Lieferung von funktionsfähiger Software zusätzlich beschleunigt wird. Zusammengefasst existieren folgende Eigenschaften / Prinzipien im Agilen Testing:

Kooperation	• Tester und Entwickler arbeiten eng zusammen und sind im gleichen Team
nicht Linear	• keine unterschiedliche Stufen für Tests und Entwicklung • Alle Testaktivitäten im Sprint fokussieren sich auf die Entwicklung
Resultatorientiert	• Keine (exzessive) Dokumentation was gemacht werden muss, sondern beschreiben, was gemacht wurde.

Abbildung 11: Grundlegende Eigenschaften im agilen Testing

Quelle: In Anlehnung an Heuer (2014, S. 9-10)

Die erwähnten Feature-Tests stehen hier etwas im Widerspruch zu diesen Aussagen. Dort existieren sehr wohl unterschiedliche Stufen für Tests und Entwicklung. Auch sind Test- und Entwicklungspersonen je nach Zusammenstellung in unterschiedlichen Teams vertreten. Ein Richtig oder Falsch ist hier jedoch schwer auszumachen. Besser wäre womöglich, die Vorteile einer engen Kooperation mit der Notwendigkeit von Feature-Tests zu kombinieren. Ein möglicher Ansatz wäre hier, die regulären Sprints für die Entwicklung zu unterbrechen und die Feature-Tests in einem separaten Sprint, durch ein erweitertes Entwicklerteam durchzuführen.

Abbildung 12: Feature-Tests als separater Sprint

Quelle: Eigene Darstellung

Um die Einbettung des Testing innerhalb der Sprints noch besser zu visualisieren, sind in Abbildung 14 die einzelnen Bestandteile eines Sprints abgebildet. Es wird ersichtlich, dass Testing integrierter Teil des Entwicklungsprozesses in jedem Sprint ist.

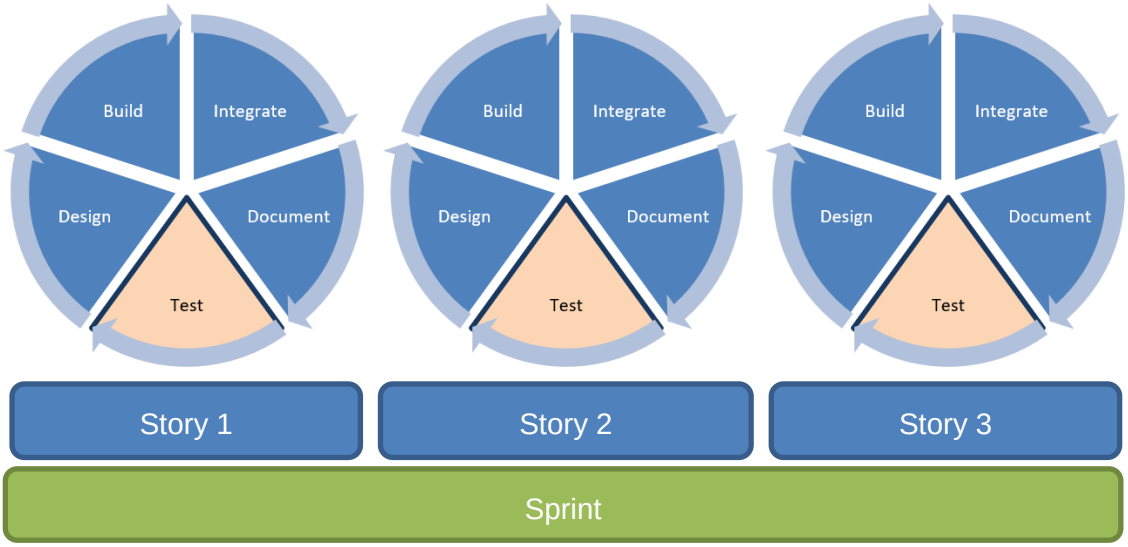

Abbildung 13: Bestandteile eines Sprints

Quelle: In Anlehnung an Heuer (2014, S. 11)

(Heuer, 2014) beschreibt, dass gefundene Fehler im Normalfall in einem zentralen Werkzeug gesammelt und dessen Behebung in demselben Sprint terminiert werden. Fehler, welche Stakeholder finden, werden im Normalfall im Folgesprint behoben (S. 11). So ist gewährleistet, dass das erstellte Produkt grundsätzlich ausgeliefert werden könnte.

2.1.5 Testmanagement vs. agile Testing

Die Rolle des Testmanagers hat sich im agilen Umfeld stark gewandelt. Im traditionellen Umfeld musste er sich durch sämtliche Unterlagen, welche im Projekt erarbeitet wurden, durcharbeiten. Auf Basis von Beispielsweise Use-Case Beschreibungen, Anforderungsdokumenten, Architekturbeschrieben, sowie Lasten- und Pflichtenheften musste sich der Testmanager mühsam und Schritt für Schritt das Testkonzept erstellen (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 113). Die gesamten Testaktivitäten wurden somit zentral gesteuert und überwacht. Bereits die Erstellung dieses Konzeptes benötigte viel Aufwand, ohne dass nur ein Teil des Systems bereits getestet und somit noch kein Nutzen generiert wurde.

Agile Testing verhält sich im Vergleich zum Testmanagement diametral anders. Viele der oben beschriebenen Fragmente existieren im agilen Umfeld nicht, respektive werden erst in den einzelnen Sprint erarbeitet und gleichzeitig umgesetzt. Aus diesem Grund existiert ein Testkonzept, wie wir es vom Wasserfall-Modell kennen, in dieser Form im agilen Umfeld nicht. Im agilen Umfeld ist zudem die Rolle als «Testmanager» sehr umstritten. In Scrum gibt es keine zentralen Personen, welche für Tests zuständig sind. Das Team an sich ist verantwortlich, dass über die gesamte Projektlaufzeit laufend getestet wird. Dies beginnt bereits bei der Erstellung der Stories, welche im Product Backlog platziert werden. Diese müssen so formuliert werden, dass sie auch testbar sind

(Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 114). Im Gegensatz zu dieser Aussage schreibt Gloger (2016) jedoch, dass die Testfälle erst im Zuge des Sprint Planning erstellt werden (S. 188). Aus Sicht des Verfassers ist jedoch eine frühe Auseinandersetzung mit dem Thema Testing wichtig und richtig. Wenn sich bereits der Product Owner mit Thema Testing befasst, leistet er wertvolle Arbeit und verkürzt so den Aufwand in den Planning Meetings, da die Stories bereits klarer und verständlicher formuliert sind.

Trotz dieser Dezentralisierung des Testmanagers zeigt sich in vielen Projekten, dass sich oft ein spezifisches Teammitglied dem Thema Qualität annimmt. Ob diese Person nun Testmanager/in oder Tester/in genannt wird, oder lediglich die Aktivitäten bezüglich Testing übernimmt, spielt grundsätzlich keine Rolle. Wichtig ist, dass ohne diese Aktivitäten der Qualität der Ergebnisse nicht genügend Rechnung getragen wird (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 114).

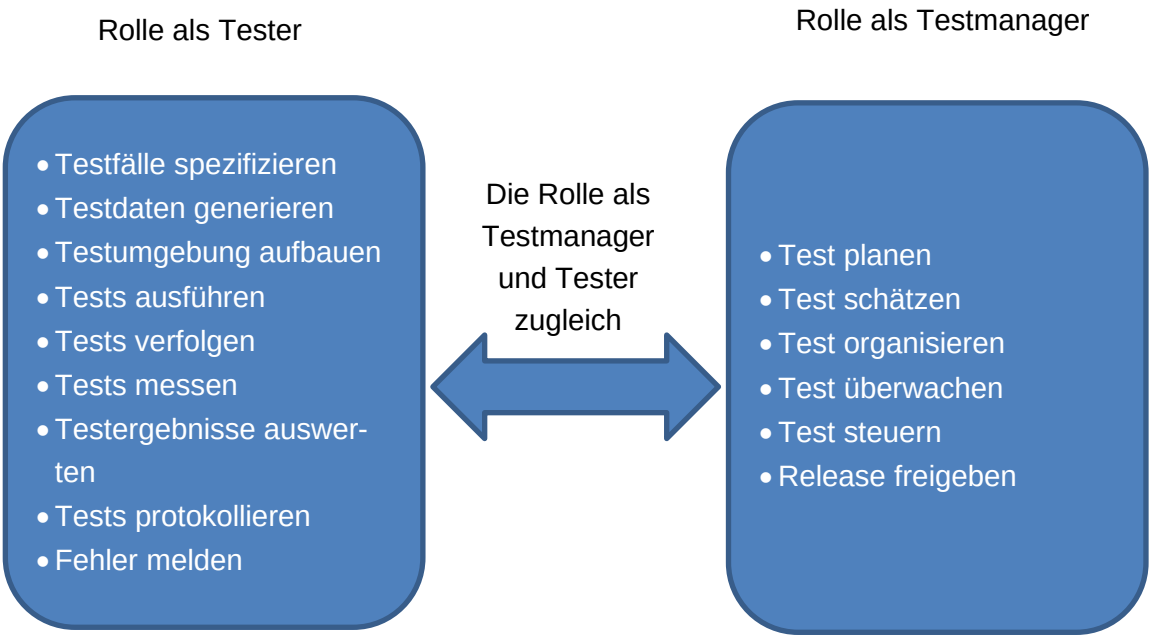

Abbildung 14: Duale Rolle als Testmanager und Tester

Quelle: In Anlehnung an Baumgartner, Klonk, Pilcher, Seidl, & Tanczos (2018, S. 112)

Bucka-Lassen (2020) vertritt hier die Meinung, dass die Entscheidung, ob interne Tester/innen eingesetzt werden, Sache des Teams sei und nicht des Managements ausserhalb des Teams.

Es ist jedoch zu beachten ist, dass ab einer gewissen Projektgrösse und Anzahl Teams das Testen innerhalb des Sprints schwieriger wird. Je länger das Projekt dauert, umso mehr erledigte Stories müssen bei den Tests, speziell bei den Integrationstests, berücksichtigt werden. So ist die Platzierung eines übergreifenden Feature-Test-Teams wie bei Abbildung 10 zu empfehlen. Dieses Team soll nicht die Testaktivitäten in den Sprints eliminieren, sondern hat den Fokus auf den erstellten Features sowie dessen

Zusammenspiel (Integrationstests). Ob diese Feature-Tests nun autonom stattfinden (Test-Team), oder als «Feature-Test-Sprint» die regulären Sprints unterbrechen (erweitertes Entwicklerteam), ist ein Testmanager oder Testmaster (SwissQ Consulting AG, 2020) für einen reibungslosen Ablauf zu empfehlen. Diese Person kann gemeinsam mit den Teams und den Stakeholdern die Testfälle auf Stufe Feature definieren und auch administrative und vorbereitende Aufgaben erfüllen. So können die Tests effizient durchgeführt werden und die Entwicklungs-Teams werden während der Sprints weitgehend entlastet.

In der Literatur herrscht durchwegs Einigkeit bezüglich des Zeitpunktes der Test. So wird generell geschrieben: «Je früher desto besser». Ritterkamp (2013) schreibt diesbezüglich: «Je später mangelhafte Software offen gelegt wird, umso aufwändiger wird es, sie zu verbessern» (S. 15). Obwohl in agilen Projekten grundsätzlich schneller Korrekturen gemacht werden können, hat diese Aussage auch heute noch seine Richtigkeit. Sobald eine Story als fertig markiert wurde, braucht es wieder einen Initialaufwand, um Fehler in einem nachgelagerten Sprint zu beheben. Es muss ein Backlog Eintrag erstellt werden, welcher bewertet und priorisiert werden muss. Ebenso muss dieser Eintrag im Sprint Planning behandelt und anschliessend im Sprint erledigt werden. Zudem könnte es im schlechtesten Fall sein, dass der Release bereits bei den Kunden ausgeliefert wurde, was einen Imageschaden sowie mögliche Hotfixes auslösen könnte.

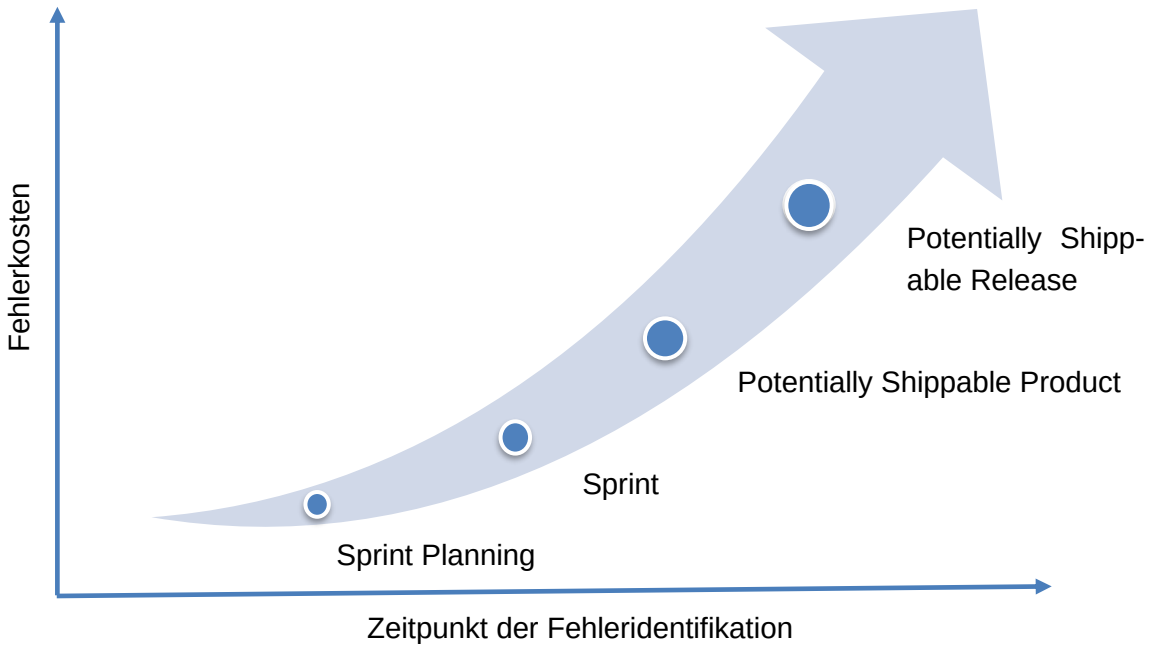

Abbildung 15: Kosten zum Zeitpunkt der Fehlerentdeckung

Quelle: Eigene Darstellung

2.2 Methoden und Techniken

In diesem Kapitel wird beschrieben, welche Methoden und Techniken im agilen Testing eingesetzt werden können. Zusätzlich wird hier auf das Themengebiet Testautomatisierung eingegangen.

2.2.1 Eignung der Tests

Gregory & Crispin (2015) beschreiben in ihrem Buch, dass Test in unterschiedlichen Dimensionen und entsprechend in vier Quadranten eingeteilt werden können (S. 122). Dabei wird zwischen den Dimensionen fachlich/technisch und teamunterstützend/produkthinterfragend unterschieden. Auf den ersten Blick fragt man sich, wieso eine solche Unterscheidung gemacht werden soll. Auf den zweiten Blick ist diese Aufteilung sehr nützlich und hilfreich, da grundsätzlich mit unterschiedlichen Sichten getestet werden kann, was auch Baumgartner, Klonk, Pilcher, Seidl, & Tanczos (2018) bestätigen (S. 47).

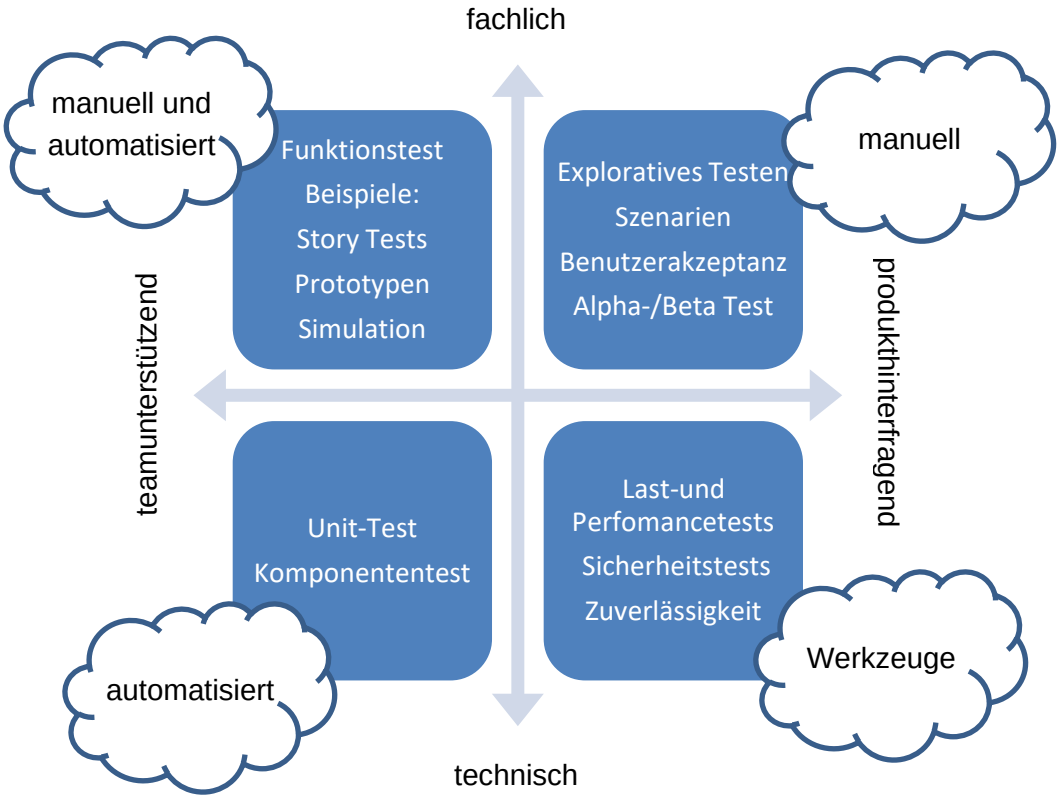

Abbildung 16: Die vier Testquadranten des agilen Testens

Quelle: in Anlehnung an Gregory & Crispin (2015, S. 122)

Die Felder auf der linken Seite sind teamunterstützende Tests und zeigen auf, ob die Tasks korrekt umgesetzt wurden. Dies sind auch oft diejenigen Testfälle, welche klar ersichtlich sind und oft aus den Storys hervorgehen. So wird sichergestellt, dass die Grundfunktionen korrekt funktionieren. Auf der rechten Seite sind Tests enthalten, welche das Produkt grundsätzlich hinterfragen respektive ob man an alles Notwendige gedacht hat (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 47). Im klassischen

Requirements Engineering könnte man unter anderem auch von implizit angenommenen Anforderungen sprechen, welche so gar nie erwähnt wurden.

Die Unterscheidung zwischen technisch und fachlich dient hauptsächlich zur Unterscheidung, wer den Fall testen kann. Wo technische Tests eher bei dem Entwicklungs-Team liegen, beispielsweise ob Werte korrekt berechnet werden und bei welcher Variante welcher Output geliefert wird, befassen sich die fachlichen Tests eher mit effektiven Prozessdaten und das Verhalten in den Prozessen respektive Teilprozessen (Gregory & Crispin, 2015, S. 122). Unter Berücksichtigung der Feature-Tests-Teams liegen die fachlichen Tests tendenziell bei dem Feature-Test-Team und die technischen bei dem Entwicklungs-Team. Jedoch ist aufgrund der gewünschten Früherkennung der Fehler empfehlenswert, dass auch das Entwicklungs-Team fachliche Tests während der Sprints durchführt, sofern das möglich ist.

2.2.2 Arten von Tests

Nachfolgend wird auf die einzelnen Arten von Test eingegangen, welche teilweise auch in der Abbildung 17: Die vier Testquadranten des agilen Testens erwähnt sind. Dies ist hilfreich um die unterschiedlichen Sichtweisen zu zeigen, welche Tester/innen im Verlaufe des Projektes einnehmen müssen. Während bei Unit-Tests der Fokus auf die kleinste Einheit geworfen wird, liegt beispielsweise bei Integrationstests der Fokus auf dem Zusammenspiel von mehreren zusammenhängenden Einheiten.

2.2.2.1 Unit-Tests

Unit-Tests sind die ersten Tests, welche in der agilen Entwicklung durchgeführt werden. Sobald in den Sprints die Entwicklung beginnt, werden die erstellten Einheiten getestet. Im Normallfall sind es die Entwickler selbst, welche diese Tests ausführen (Heuer, 2014, S. 34). Unit-Tests sind grundsätzlich unabhängig voneinander. Es wird lediglich diejenige Funktionalität getestet, welche der Entwickler in der aktuellen Aufgabe umgesetzt respektive programmiert. Beispiel: Der Entwickler programmiert eine Benutzeroberfläche für die Erfassung eines neuen User-Accounts. Er testet anschliessend, ob bei den Feldern die gewünschten Zeichenformate und Zeichenfolgenlängen eingegeben werden können. Grundsätzlich erfüllen die Summe aller Unit-Tests die Akzeptanzkriterien einer User Story.

Wenn zu diesem Zeitpunkt Fehler entdeckt werden, können sie sofort durch den Entwickler behoben werden. Wird zu diesem Zeitpunkt nicht oder nachlässig getestet, kann dies erhebliche Auswirkungen auf die nachfolgenden Funktionalitäten haben und entsprechend hohe Kosten verursachen.

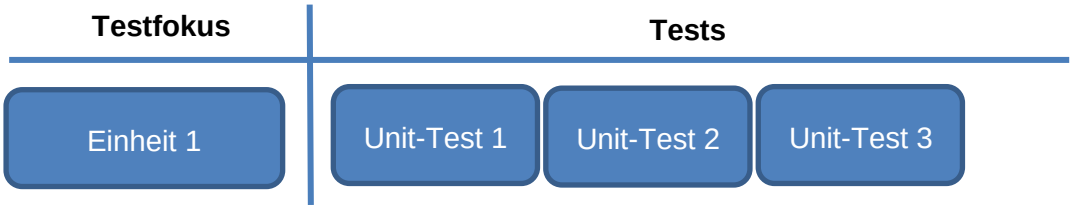

Abbildung 17: Unit-Tests

Quelle: Eigene Darstellung

2.2.2.2 Komponenten / Integrationstest

Komponententests können als eine Vorstufe von Integrationstests angeschaut werden (Heuer, 2014, S. 35). Bei Komponenten- oder Integrationstests werden mehrere entwickelte Einheiten in Ihrem Zusammenspiel getestet. Es wird beispielsweise getestet, ob der User-Account eröffnet werden kann, und gleichzeitig auch ein Passwort gesetzt wird. Zusätzlich wird geprüft, ob bei dem User Account mehrere Liefer- und Rechnungsadressen hinzugefügt werden können. So bekommen Tester/innen eine andere Sicht auf die Funktionalität. Der Testfokus entfernt sich von isolierten Einheiten, hin zum Zusammenspiel mehrerer Einheiten. Je nach Storygrösse liegt der der Komponententest noch beim Entwicklungs-Team. Integrationstest sind dann eher in Richtung Endbenutzer / Feature-Test-Teams anzusiedeln. Im Idealfall werden Fehler auf Stufe Komponente noch im Entwicklungs-Team entdeckt. So können diese noch in demselben Sprint behoben werden. Wird der Fehler erst später entdeckt, ist der Aufwand entsprechend höher.

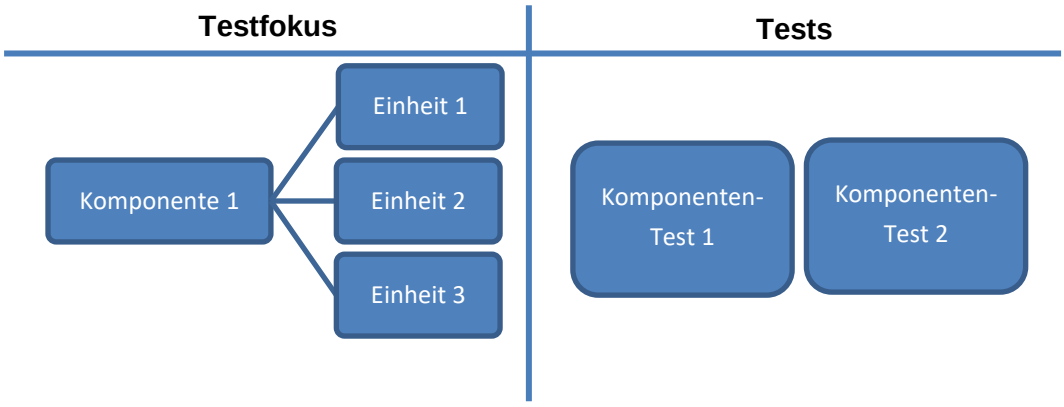

Abbildung 18: Komponententests

Quelle: Eigene Darstellung

2.2.2.3 Feature-Tests / Systemtests

Feature- oder Systemtests sind die nächsthöhere Stufe der Tests. Hier ist der Testfokus auf die Gesamtfunktionalität gelegt. Konkret wird hier das Zusammenspiel sämtlicher Komponenten getestet. Beispielsweise wird hier geprüft, ob sich der Benutzer anmelden kann, Produkte dem Warenkorb hinzufügen kann, seine vorab erfasste Rechnung und Lieferadresse wählen und die Bestellung abschliessen kann. Zu diesem Zeitpunkt wurden die Komponenten im Gesamtsystem integriert. Werden hier Fehler entdeckt, ist der Aufwand für die Korrekturen grösser, als wenn sie bei den Unit- oder Komponententests entdeckt werden. Auch die Identifikation der Ursache ist im Zusammenspiel der Units schwieriger. In der agilen Entwicklung sollten die Einheiten und Komponenten so schnell als möglich in die Integrationsumgebung eingespielt und laufend getestet werden (wenn möglich automatisiert) (Heuer, 2014, S. 34). So bekommt das Entwicklungs-Team schnell Feedback betreffend den Test und kann die Fehler im Idealfall noch im laufenden Sprint beheben. Sofern nur ein Team an der Entwicklung des Produktes arbeitet, laufen Integrationstest oft relativ reibungslos ab. Sobald jedoch mehrere Teams involviert sind, wird die Abstimmung und Testkoordination schwieriger und es werden aufwändigere Test notwendig (Heuer, 2014, S. 34).

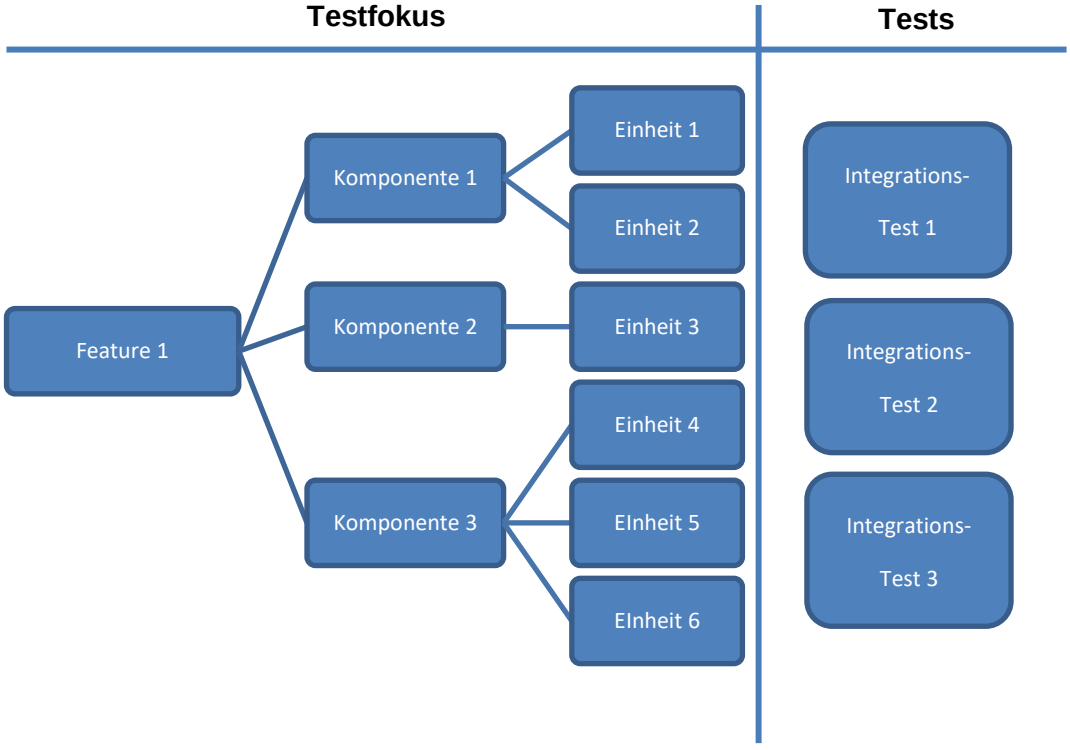

Abbildung 19: Feature-Tests

Quelle: Eigene Darstellung

2.2.2.4 Systemabnahmetest

Systemabnahmetests sind die finalen Tests, bevor ein Release freigegeben wird. Zu diesem Zeitpunkt sollten Fehler die Ausnahme sein. Im Grundsatz unterscheiden sich die Abnahmetests nicht von den Unit- oder Feature-Tests. Vielmehr ist die Summe aller erstellten und bestandenen Tests Grundlage für die Systemabnahmetests (Gregory & Crispin, 2015, S. 124). Heuer (2014) empfiehlt hier ein separates Testteam zu bilden, das einen grossen Teil der Arbeit abnimmt und zusätzliches Knowhow einbringt (S. 37). Diese Aussage deckt sich mit der Installation eines Feature-Test-Teams, wie es SwissQ Consulting AG (2020) in Ihrem Test-Framework empfiehlt. Jedoch wird in diesem Fall das Feature-Test-Team wesentlich früher in den Testaktivitäten eingesetzt, als bei der Systemabnahme.

2.2.3 Methoden

In der Praxis haben sich unterschiedliche Test-Methoden etabliert, welche in den nachfolgenden Kapiteln beschrieben werden.

2.2.3.1 Test Driven Development

Test Driven Development, oder abgekürzt "TDD", hat sich mittlerweile in vielen agilen Teams zur Selbstverständlichkeit entwickelt und geniesst unter IT-Experten viel Anerkennung (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 49). Auch Heuer (2014) erwähnt, dass sich TDD der agilen Entwicklung de-facto-Standard entwickelt hat (S. 18). Bucka-Lassen (2020) hingehen widerspricht teilweise diesen Aussagen. So stecke Testautomatisierung, wie es TDD vorsieht, «in vielen traditionellen Unternehmen (z.B. Banken oder Versicherungen) noch in den Kinderschuhen». Daraus könnte die Vermutung aufgestellt werden, dass Unterschiede bestehen zwischen Unternehmen, deren Kerngeschäft die Softwareentwicklung ist und Unternehmen, welche Software lediglich als Werkzeug für ihr Kerngeschäft nutzen.

Im TDD Ansatz werden Testfälle ganz zu Beginn geschrieben (programmiert), bevor mit der effektiven Entwicklung begonnen wird (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 47). Dies hat den wesentlichen Vorteil, dass bereits vor der Entwicklung mögliche Testszenarien überlegt werden und entsprechend bereits mit dem korrekten Fokus der Code erstellt wird. Durch die vorgängig erstellten Testfälle werden die Anforderungen respektive Stories in wesentlichem Masse verfeinert, und gleichzeitig können dieselben im Testing verwendet werden. Rupp (2014) beschreibt ebenfalls, dass auch beim klassischen Requirements Engineering früh erstellte Testfälle die Qualität der Anforderungen wesentlich erhöht (S. 323).

Zusammengefasst kann die Schlussfolgerung gezogen werden, dass sich TDD, zumindest im Bereich Softwareentwicklung, weiterentwickeln und in hohem Masse Erfolgsversprechend ist. Im Kontext zu einem ERP-System ist TDD jedoch mutmasslich weniger geeignet, da in diesen sehr wenig programmiert wird. Entsprechend ist das Potential von TDD wohl eher gering. Auch Meier (2020) kennt ein solches Vorgehen bei seinen

Kunden nicht. Was jedoch für ein ERP-Projekt ebenfalls wertvoll sein kann, ist die vorgängige Erstellung der Test (sofern sie notwendig sind) welche über die Akzeptanzkriterien hinausgehen. Ob diese dann programmiert werden oder nicht, hängt von dem Inhalt der Story sowie den zur Verfügung stehenden Werkzeugen ab.

TDD ist im Testquadrant unten links respektive als technisch orientiert und teamunterstützend anzusiedeln (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 47). Die Testfälle sind in ihrer Ausprägung sehr technisch formuliert, und wenig auf den Gesamtprozess ausgelegt. TDD wird somit häufig bei Unit-Tests eingesetzt (Heuer, 2014, S. 34). Beispielsweise werden in solchen Tests unterschiedliche Berechnungen modelliert oder mögliche Zielausprägungen von Zuständen beschrieben. Diese Tests werden in der Praxis so weit als möglich automatisiert. Die Entwickler programmieren so lange, bis der Code den Testfall «besteht» (Heuer, 2014, S. 18). Dies hat den wesentlichen Vorteil, dass durch die stetige Weiterentwicklung die neu erstellten Codes ständig und mit wenig Aufwand übergeprüft werden können. Aus Sicht der Gesamtintegration der Komponenten ist dies unerlässlich und bringt in späteren Testphasen einen wesentlichen Mehrwert. (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 48).

Der Ablauf von TDD sieht zusammengefasst wie folgt aus:

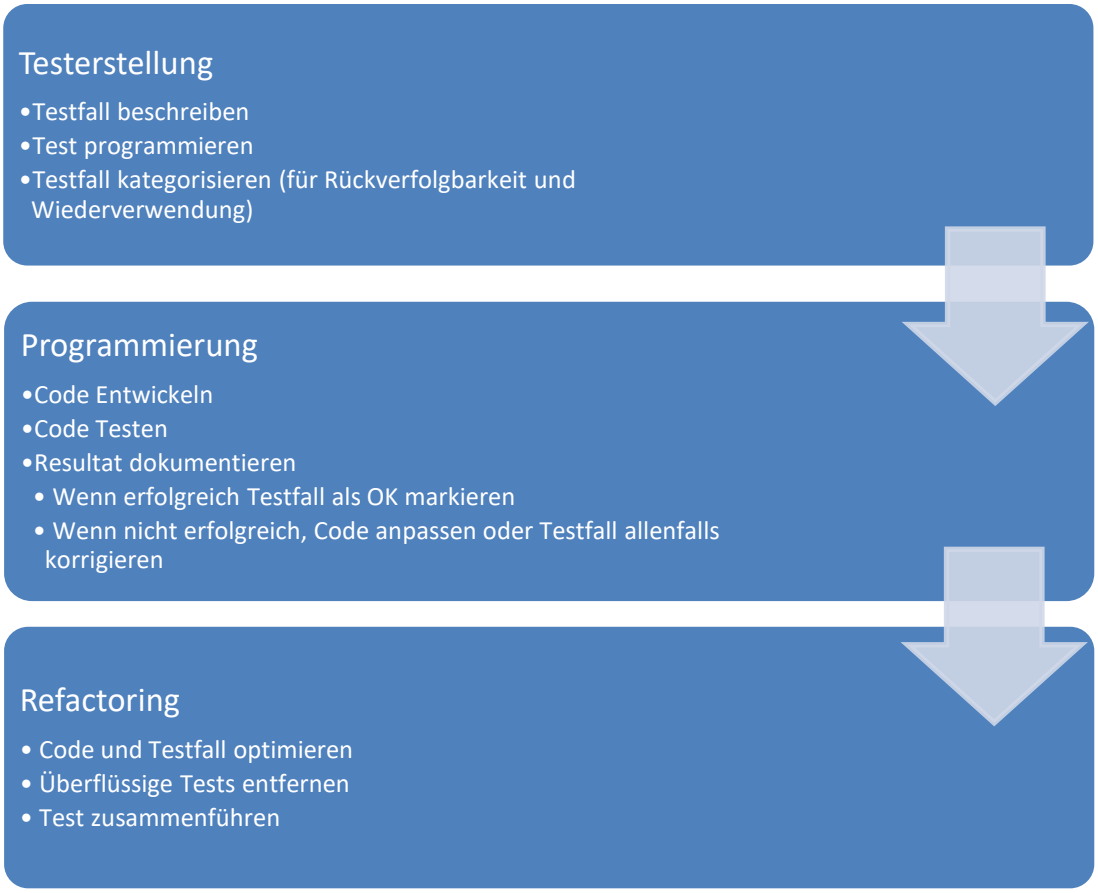

Abbildung 20: Ablauf Test Driven Development

Quelle: In Anlehnung an Baumgartner, Klonk, Pilcher, Seidl, & Tanczos (2018, S. 49)

2.2.3.2 Acceptance Test Driven Development und Behavior Driven Development

Bei Acceptance Test Driven Development (ATDD) und Behavior Driven Development (BDD) oben links respektive als fachlich orientiert und teamunterstützend anzusiedeln. Während sich TDD auf die einzelne Komponente, respektive Unit fokussiert, also sehr technisch orientiert, wird bei ATDD und BDD das Zusammenspiel der Komponenten und Units geprüft (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 51). Beispielsweise wird hier nicht nur getestet ob bei einer Lagerentnahme die korrekten Mengen am korrekten Lager abgebucht wurden, sondern auch, ob gleichzeitig ein Lieferschein generiert wurde und ob die entnommenen Mengen auch bei der Kundenbestellung als geliefert ersichtlich sind. Bei solchen Tests ist es wichtig, die Reihenfolge der Entwicklung im Sprint zu koordinieren, wie es auch bei einem Release-Backlog der Fall wäre. Nur so können fachliche Tests, welche über mehrere Units laufen, frühzeitig getestet werden (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 51). Baumgartner (2018) beschreibt, dass solche Tests währen der Entwicklung regelmässig wiederholt werden sollen, damit die Funktionalitäten dauerhaft sichergestellt werden können (S. 52).

Bei ATDD werden ausführbare Abnahmetest anhand von Beispielen von Kunden entwickelt. Jedes Beispiel wird besprochen und wenn möglich programmiert. Auf Basis dieser Test werden die Funktionalitäten entwickelt. Sobald der Test erfolgreich durchgeführt wurde, kann dieser potentiell als automatisierter Regressionstest verwendet werden (Gregory & Crispin, 2015, S. 819).

BDD ist eine Variation von TDD. Dabei wird die Spezifikation in natürlicher Sprache (domänenspezifische Sprache) geschrieben. Die Testfälle müssen jedoch von Personen verfasst werden, welche die domänenspezifische Sprache anwenden können. Für Kunden ist es jedoch einfacher, diese Spezifikation zu lesen und zu verstehen. BDD-Testfälle können relativ einfach automatisiert werden (Gregory & Crispin, 2015, S. 174).

In ihrer Ausprägung sind ATDD und BDD Tests, welche bei Komponenten- und Integrationstests anzusiedeln sind. Aus Sicht des Verfassers stellt sich hier die Frage, ob das Entwicklungs-Team aus zeitlichen Gründen diese Tests selbst erstellen und durchführen kann. Sofern die Testautomatisierung betrieben wird, sind diese Tests sicherlich beim Entwicklungs-Team anzusiedeln. Sofern jedoch keine Testautomatisierung stattfinden, werden diese Tests auf dieser Stufe schnell aufwändig und müssten tendenziell durch ein separates Team durchgeführt werden.

2.2.3.3 Technisch orientiert und produkthinterfragend

Die Tests in diesem Quadranten werden oft nicht durch die agilen Teams durchgeführt. Es handelt sich hier um Spezialtests, welche meist von Expert/innen und spezifischen Tools durchgeführt werden (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 54). Beispiele für solche Test können sein:

- Last- und Perfomance Test
- Sicherheitstests
- Zuverlässigkeitstests

Im Kontext zu einer ERP-Einführung hat sich in der Praxis gezeigt, dass solche Tests eher selten durchgeführt werden. Im Normalfall werden die Systeme aufgrund Vorgaben Seitens der ERP-Anbieter aufgesetzt, was betreffend Zuverlässigkeit und auch Sicherheit bereits eine breite Abdeckung beinhaltet. Zudem sollte jedes Unternehmen Sicherheitsrichtlinien besitzen, welche den Zugriff auf Applikationen und Daten generell regelt und entsprechend nur berechtigten Personen den Zugriff ermöglicht. Ganz ausser Acht lassen sollte man diese Tests jedoch nicht. «blindes Vertrauen» kann auch erhebliche Überraschungen hervorbringen. Oft werden Mängel bezüglich Zuverlässigkeit auch indirekt bei Test in anderen Quadranten bemerkt.

Das Testen von Last- und Performance ist generell schwierig, ohne in den Produktivbetrieb zu wechseln. So müssten sich nahezu gleich viele User wie im Produktivbetrieb auf dem Testsystem anmelden und gleichzeitig Rechenleistung beanspruchen. Mit «echten Anwendern» ist dies schwer lösbar. Sofern für solche Tests eine Infrastruktur bereitstehen würde, wäre der Aufwand nicht unerheblich solche Tests durchzuführen. Jedoch soll auch hier währen den normalen Testaktivitäten, speziell auch während den Feature-

Tests, ein Augenmerk auf die Performance gelegt werden. Sind die Reaktionszeiten während der Tests bereits hoch, sind das Anzeichen für ein Problem.

2.2.3.4 Exploratives Testen

Explorative Tests sind klassische Endbenutzer-Tests. Sie sind im Testquadrant oben rechts, fachlich orientiert und produkthinterfragend anzusiedeln. Bei diesen Tests ist die Intuition des Endbenutzers gefragt (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 54). Hier werden grundsätzlich keine Testspezifikationen geschrieben, sondern es wird «munter darauf los getestet». So bewegt sich der Anwender relativ frei im Rahmen der bereit erstellten Funktionalitäten und reagiert auf gefundene Fehler. So können Fehler gefunden werden, an welche bei den Testspezifikationen nicht gedacht wurde. Baumgartner (2018) beschreibt, dass dieses Explorative Testen jedoch schnell unkontrolliert und unkoordiniert verlaufen könne. So wird empfohlen, unterschiedliche Testsessions zu machen, bei welchen die gefundenen Fehler besprochen werden konkrete Tests für nachfolgende Sessions definiert werden (S. 54).

2.2.4 Testautomatisierung

Jeder der in Projekten bereits in der Funktion als Tester/in mitgearbeitet hat, kennt den verlockenden Ruf der «Testautomatisierung». Wie schön wäre es, wenn derselbe Testfall nicht immer wieder manuell erstellt werden müsste. Der Schreibende selbst hatte diese Erfahrung gemacht. In einem Projekt für Digitalisierung des Kreditoren-Workflows wurde eine Applikation beschafft, mit welcher Kreditoren-Rechnungen eingelesen und von den Bestellern und visumsberechtigten Personen digital genehmigt werden konnten. Dazu mussten viele unterschiedliche Tests für Bestell- und Rechnungskonstellationen erstellt und verbucht werden. Wie in vielen Wasserfall-Projekten üblich, scheiterten bereits die ersten Testfälle aufgrund von Fehlern in der Schnittstelle. Nach und nach wurden diese Fehler behoben. Jedoch mussten die Testfälle nach jedem Entwicklungsschritt wieder erneut erzeugt werden. Auch bereits erfolgreich durchgeführte Tests mussten in einem reduzierten Rahmen in Form von Regressionstest erneut erstellt und durchgeführt werden. Die Entwickler konnten aufgrund Anpassungen der Schnittstelle nicht sicher gehen, dass die bereits getesteten Funktionalitäten weiterhin stabil laufen. Über die gesamte Entwicklungszeit haben wir Wochen verbracht mit der Erstellung und Durchführung von Tests.

In agilen Projekten verstärkt sich dieser Aufwand noch, da die Entwicklungszyklen kürzer sind und die Applikation laufend und über die gesamte Laufzeit verändert wird. So wird der Ruf nach Testautomatisierung-Werkzeugen noch lauter. «Jeder, der das Testen in agilen Projekten einigermassen beherrschen will, kommt daher nicht um ein Werkzeug herum» (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 159). Auch Heuer (2014) schreibt, dass bei agilen Methoden sehr viel getestet werden muss und aufgrund Zeit- und Kosteneinsparungen so viele Tests wie möglich automatisiert werden sollen (S. 31).

Nun könnte man daraus schlussfolgern, dass zwingend ein Testautomatisierungs-Werkzeug angeschafft und flächendeckend eingeführt werden soll. Dies würde dem agilen ERP-Projekt massiv an Zeit einsparen und zudem auch bei anderen Applikationen verwendet werden können. Genau dieser Schlussfolgerung wird jedoch widersprochen (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 159). Viel mehr wird empfohlen, dass Testwerkzeuge in einem kleinen Rahmen in Teams pilotiert werden sollen (Bucka-Lassen, 2020). Zudem sind für den Einsatz von Testwerkzeugen Testexperten notwendig. Es wird davor gewarnt, klassische Automatisierungswerkzeuge in der agilen Entwicklung einzusetzen. Der Einführungs- und Schulungsaufwand ist oft sehr umfangreich. Mit etablierten Werkzeugen kann oft nicht genug schnell auf Veränderungen der Software reagiert werden. Aus diesem Grund wird eher empfohlen, auf schlankere Testautomatisierungswerkzeuge, welche sich besser für Agile Entwicklungen eignen, zurückzugreifen (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 176). Ritterkamp (2013) empfiehlt, dass «viele kleine und automatisierte Tests die Anzahl von komplexen Tests massiv reduzieren könne» (S. 15).

Weiter ist zu beachten, dass Testautomatisierung nicht in jedem Fall geeignet ist. In Kapitel 2.2.1 Eignung der Tests wird beschrieben, dass technisch orientierte und teamunterstütze Test vor Allem für die Automatisierung geeignet sind. Ergänzend dazu hat Heuer (2014) folgende Auflistung für die Eignung von automatisierten Tests erstellt (S. 32):

Phase	**Task**	**Automatisierung sinnvoll?**
Sprint 0	Anforderungen	Nein
	Design	Nein
Sprint 1-X	Code Überprüfung	Ja
	Unit-Test	Ja
	Integrationstest	Ja
	Akzeptanztests	Nein
Auslieferung	Test nach Auslieferung	Nein

Tabelle 2: Eignung von Testautomatisierung

Quelle: Heuer (2014, S. 32)

Gregory & Crispin (2015) erwähnen in Ihrem Buch die Testpyramide, welche Sie für unterschiedliche Sichten auch erweitert haben. In der originalen Pyramide zeigen sie auf, welche Test auf welchen Stufen den höchsten Nutzen bei Testautomatisierung bringen (S. 678). So sind Unit- und Komponententest grundsätzlich für das Team die wertvollsten für eine Automatisierungen.

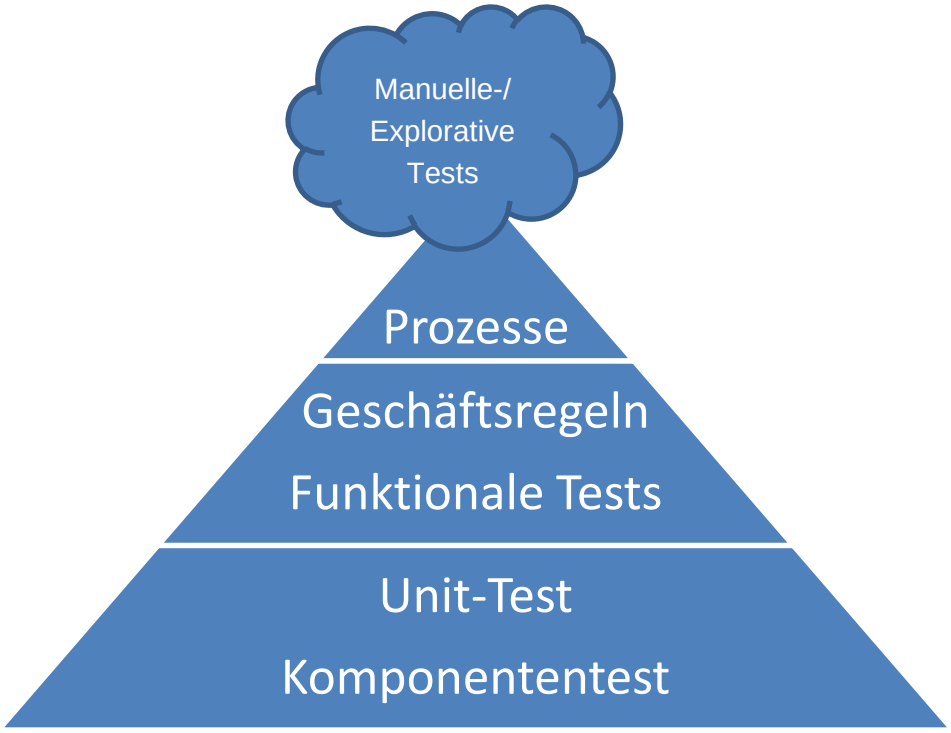

Abbildung 21: Testautomations-Pyramide

Quelle: Gregory & Crispin (2015, S. 678)

Wenn wir diese Expertenmeinungen zusammenfassen, können wir nun folgende Schlussfolgerungen ziehen:

- Für die Anwendung von Testwerkzeugen sind Experten notwendig, welche die Tests schreiben / erstellen können
- Je tiefer die Testebene, desto höher ist der Nutzen von Automatisierung
- Es ist erfolgsversprechender, kleinere und flexiblere Testwerkzeuge einzusetzen als komplizierte Werkzeuge mit grossem Funktionsumfang
- Kleine Automatisierungen sind effizienter als komplizierte und aufwändige Testfälle
- Die Pilotierung von Testwerkzeugen in kleinen Gruppen ist erfolgversprechender als die flächendeckende Einführung von mächtigen Werkzeugen

Wenn wir aufgrund dieser Aussagen nochmals die Tests in dem Projekt «Digitalisierung Kreditoren-Workflow» betrachtet werden, hätte bereits die automatisierte Anlage von Bestellungen und Rechnungen einen grossen Teil des Testaufwandes eliminiert. Es hätte nach jeder Anpassung jeweils ein neues Set an Bestellungen automatisch generiert

werden können. So wären die Tester/innen nicht mit dem generieren von Stammdaten beschäftigt gewesen, sondern effektiv mit der Ausführung der Test und der Interpretation der Resultate. Der Aufwand für die komplette Automatisierung sämtlicher Workflow-Konstellationen wäre vermutlich nur mit grossem Aufwand zu bewältigen gewesen. So wäre mutmasslich das Testen der Testautomatisierung aufwändiger gewesen als eine manuelle Durchführung der Tests.

2.3 Dokumentation

Die Dokumentation von Software ist sowohl in agilen als auch in Wasserfall-Projekten ein umstrittenes Thema und wird in vielen Fällen vernachlässigt. In agilen Projekten neigen Entwickler generell dazu, wenig bis gar keine Dokumentation zu erstellen.

Einer der Grundsätze von Agilität ist «Funktionierende Software hat Vorrang vor umfassender Dokumentation» (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 141). Dieser Grundsatz wird oft als Vorwand für fehlende Dokumentationen verwendet. Auch in klassischen Projekten kommt die Dokumentation oft zu kurz. Ein häufiger Grund dafür ist, dass am Ende des Projektes das Budget bereits überschritten wurde und der Druck nicht allzu hoch ist, für die erstellte Software noch eine ausführliche Dokumentation zu erstellen. Auch in der Weiterentwicklung (Change Request) von Software werden Dokumentationen oft vernachlässigt. Sind Kunden zufrieden und das System funktioniert einwandfrei, besteht von Seiten der Anwender häufig kein Bedürfnis nach einer Dokumentation. Dass diese Nachlässigkeiten später negative Folgen haben kann, wurde schon in vielen Projekten festgestellt.

Solange ein System einwandfrei funktioniert und nicht verändert wird, braucht es grundsätzlich auch keine Dokumentation. Sobald jedoch beispielsweise ein Data Scientist Auswertungen machen möchte, und das Datenmodell nicht dokumentiert ist, muss sich dieser zuerst aufwändig einen Überblick verschaffen. Auch beispielsweise bei Anpassungen oder Umstellungen von Schnittstellen sind fehlende Beschriebe der Funktionalitäten und Logiken verehrend. Schon oft mussten wir bei Anpassungen den Code studieren oder mit Datenanalysen die programmierte Logik herausfinden. Wir verwenden für diese Art von Requirements Engineering spasseshalber den Ausdruck «System Forensik».

Wenn nun das Thema Dokumentation mit Fokus auf die Einführung eines ERP-Systems betrachtet wird, wird schnell deutlich, dass eine Dokumentation benötigt wird. Auf der einen Seite besteht das Bedürfnis von Seiten der Anwender, die unterschiedlich geschickt mit Systemen umgehen können. Für diese Anwender ist es wesentlich, eine Dokumentation oder ein Handbuch zur Seite zu haben, wenn sie mit dem System zu arbeiten beginnen. Auf der anderen Seite müssen Tester/innen aus den Fachbereichen Soll-Ist-Vergleiche der Funktionalitäten durchführen können (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 141). Was mit Sicherheit gesagt werden kann: Die wenigsten von diesen beiden Anspruchsgruppen sind in der Lage «System Forensik» zu betreiben.

An dieser Stelle ist es wichtig zu verstehen, welche Dokumente in einem Projekt entstehen und zu welchem Zweck diese Dokumente dienen. Vor allem in agilen Projekten sollte Wert daraufgelegt werden, Dokumente nicht zum Selbstzweck zu erstellen. «So viel wie nötig, so wenig wie möglich» muss hier als Leitsatz dienen (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 143). So ist es sinnvoll, sich zu Beginn des Projektes mit den Dokumenten und Anspruchsgruppen auseinander zu setzten. Aufgrund der engen Zusammenarbeit in agilen Teams könne jedoch gemäss Ritterkamp (2013) auf eine strikte und formalisierte Form der Anforderungs- und Testdokumentation verzichtet werden (S. 11). Es stellt sich jedoch die Frage, was geschieht, wenn mehrere Teams an einem Produkt arbeiten, oder sogar ein Feature-Test-Team involviert ist? Vermutlich wird, je grösser das Projekt ist und desto mehr Teams an einem Produkt arbeiten, die Notwendigkeit von Formalisierung grösser. Was in einem Team noch relativ einfach und unkompliziert gehandhabt werden kann, bedingt bei mehreren Teams einen höheren Koordinations- und Kommunikationsaufwand. So ist mindestens auf Stufe Feature-Tests eine formalisierte Dokumentation von Testfällen und Ergebnissen sinnvoll.

2.3.1 Storyboard

Eine geeignete Möglichkeit für die Auseinandersetzung mit Dokumentationen ist, ein so genanntes «Storyboard der Dokumentationen» zu erstellen. Dieses kann in den Sprints aktiv verwendet und auch laufend angepasst werden kann (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 143). In der Tabelle 3 exemplarisch aufgelistet, welche Dokumente und Informationen von welchen Anspruchsgruppen benötigt werden.

		Erstellung	**Wartung**	**Betrieb**
Anwender	benötigt			Handbuch Online Hilfe Hilfetexte
	erstellt			Fehlermeldungen
Entwickler	benötigt	Stories Anforderungen Product Backlog Vorgaben Fehlerbeschreibungen Architektur	Stories Anforderungen Product Backlog Vorgaben Fehlerbeschreibungen Architektur	
	erstellt	Inline Dokumentation Unit-Tests	Inline Dokumentation Unit-Tests	

		Erstellung	Wartung	Betrieb
Product Owner	benötigt	Anforderungen der Fachbereiche Lastenheft		Benutzer Feedback
	erstellt	Stories Anforderungen Product Backlog Entscheidungen Begründungen		Product Backlog
Tester/innen	benötigt	Stories Anforderungen Product Backlog Fehlerbeschreibungen Fehlerbehebungen	Product Backlog Stories Changes Fehlerbehebungen Regressionstests Testdatenbeschreibungen	
	erstellt	Testfälle Konkretisierung von Anforderungen Fehlerbeschreibungen Retest-Ergebnisse	Testfälle (neu) Konkretisierung von Anforderungen (neu) Fehlerbeschreibungen (neu) Retest-Ergebnisse	

Tabelle 3: Storyboard der Dokumentation

Quelle: In Anlehnung an Baumgartner, Klonk, Pilcher, Seidl, & Tanczos (2018, S. 144)

Die Auseinandersetzung mit der Thematik Dokumentation hat zudem noch weitere Vorteile. Ist bereits zu Beginn des Projektes klar, welche Dokumente zu erstellen sind, können diese auch strukturierter erstellt und gesammelt werden. Vor allem wenn mehrere Scrum-Teams an einem Projekt arbeiten, sollten die Dokumentationen einheitlich und wenn möglich zentral verfügbar sein. So kann diesbezüglich bereits zu Beginn die Effizient verbessert werden.

2.3.2 Von der User Story zum Feature-Test

Wie bereits im Kapitel 2.1.2 beschrieben, werden in den Sprint Planning Meetings aus User Stories Akzeptanzkriterien abgeleitet. Diese werden so lange verfeinert und

diskutiert, bis das Team die konkreten Aufgaben kennt, welche im nächsten Sprint erledigt werden müssen. Sofern die Stories genügend klein «geschnitten» sind, braucht es für die Umsetzung der Story nicht mehr viele zusätzliche Tests. Meist sind die zu Beginn formulierten Testfälle jedoch nicht in der Granularität, in welcher konkrete Tests durchgeführt werden können. So ist es durchaus sinnvoll, dass bevor die Entwickler mit ihrer Arbeit beginnen, konkrete und präzise Testfälle erstellt werden. Diese können sowohl von den Entwicklern, als auch von den Anwendern, Business Analysten oder anderen Teammitgliedern stammen. Diese Testfälle verschaffen einiges an Klarheit und verbessern die Story erheblich (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 148).

Etwas anspruchsvoller sind die Nachfolgenden Integrations- und Feature-Tests. Das Zusammenspiel von einzelnen Stories, welche teilweise sogar in unterschiedlichen Sprints erstellt wurden, muss getestet werden. Die Erstellung solcher Test Szenarien ist spezifisch in einem ERP-Projekt mitunter der höchste Aufwand (Meier, 2020). Auch Frei (2020) hat die Erfahrung gemacht, dass die Definition von Abläufen und Test Cases sehr aufwändig ist. Erschwerend kommt hinzu, dass Mitarbeitende, welche im Entwicklungs-Team beteiligt sind, vermutlich eher feature- als prozessorientiert denken. Am Ende sind es bei einem ERP-System jedoch die Prozesse, die in ihrem Zusammenspiel funktionieren müssen. All diese Faktoren machen Integrations- und Feature-Tests innerhalb des Entwicklungs-Teams schwierig, bei grossen Projekten tendenziell unmöglich.

Der Release Backlog, respektive der Release Plan, ist sicherlich ein Hilfsmittel, um die Tests entsprechend zeitlich zu planen und zu Koordinieren. Auch die Definition der Features, respektive die Zuordnung der Stories zu einzelnen Features, hat einen wesentlichen Einfluss auf diese Tests. Es ist einerseits darauf zu achten, dass alle erledigten Stories in ihrem Zusammenspiel getestet werden und auch die Prozesse in all ihren Ausprägungen und Variationen. In Hinblick auf die Feature-Tests wäre es natürlich ideal, wenn Features sehr nahe an den Prozessen definiert werden. Dadurch kann die Komplexität reduziert werden.

2.3.3 Beschreibung, Durchführung und Fehlerdokumentation

Bei agilen Vorgehensweisen wird i.d.R. mehr Zeit in die Entwicklung und Umsetzung der Produkte investiert, als in deren Dokumentation. Dies sollte auch im agilen Testing nicht anders sein. Jedoch sollten Informationen, die zu einem späteren Zeitpunkt wieder benötigt werden, auch verständlich dokumentiert werden. Baumgartner (2018) erwähnt hier explizit Retests und Regressionstests (S. 149). Im Kontext zu ERP-Systemen sind hier noch weitere Einflussfaktoren, welche zu einem späteren Zeitpunkt relevant sein könnten. So werden vom ERP-Systemanbieter laufend Systemupdates ausgeliefert, welche in einem gewissen Rahmen auch wieder getestet werden müssen (mehr dazu in Kapitel 3.2). So empfiehlt es sich, spezifische Testfälle, welche potentiell später wiederverwendet werden sollen, ausreichend zu dokumentieren. So könnten repräsentative Testfälle bei einer Übergabe des Systems an den Betrieb übergeben werden.

Betreffend Dokumentation der Testfälle und auch deren Ergebnisse ist es in der Praxis üblich, bei grösseren Projekten geeignete Werkzeuge zu nutzen. So können sowohl

Tests als auch Testergebnisse protokolliert, priorisiert und rückverfolgt werden (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 150). Dies ist mit Sicherheit auch bei der Einführung von ERP-Systemen hilfreich, da oft auch über diverse Schnittstellen mit Umsystemen kommuniziert wird und die Komplexität dabei nicht unerheblich ist. Die Dokumentation in Microsoft Word oder Excel wird in ERP-Projekten noch oft praktiziert (Meier, 2020) und ist bei den Anwendern sehr beliebt. Dies ist in den letzten Jahren durch die Office 365 Plattform noch wesentlich einfacher geworden, da neu sehr stark auf Kollaboration gesetzt wird. Trotz dieser Tatsache ist es empfehlenswert, die Vor und Nachteile eines professionellen Testwerkzeuges abzuwiegen. Die Erfahrung hat gezeigt, dass Projektteams mit MS Office Produkten bei der Anforderungs- und Testdokumentation schnell an ihre Grenzen stossen. Frei (2020) erwähnt zu diesem Punkt, dass das Werkzeug eher zweitrangig ist, wichtiger sei, dass die Personen gewohnt sind damit zu arbeiten.

Zu unterscheiden gilt es hier Tests innerhalb der Entwicklungs-Teams und Feature-Tests. Innerhalb der Sprints können die Teams relativ eigenständig testen. Auch die Dokumentation kann hier sehr einfach gehalten werden. Denn am Ende sind die Akzeptanzkriterien erfüllt und die Story erledigt, oder eben nicht. Einzig bei der Übergabe relevanter Testfälle zu den Feature-Tests ist etwas mehr Kommunikation und Formalität sicherlich hilfreich.

Feature-Tests hingegen sind repräsentativ für die Qualität der Software im Ganzen und müssen strukturiert erfasst und nachverfolgt werden. Bei der Durchführung der Tests reichen oft einfache Metriken, welche bereits eine gute Übersicht über den Stand der Tests liefern. So kann beispielsweise die Anzahl von Testfällen mit der Anzahl durchgeführter Testfälle (aufgeteilt in erfolgreich oder nicht erfolgreich) verglichen werden (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 150). So kann schnell ein Überblick über den Fortschritt der Tests sowie der Qualität der Software gewonnen werden. Gefundene Fehler müssen zwingend dokumentiert werden. So kann der Product Owner die Behebung des Fehlers in den Backlog aufnehmen und in einem der Folgesprints beheben lassen (Baumgartner, Klonk, Pilcher, Seidl, & Tanczos, 2018, S. 151). In der Praxis hat es sich gezeigt, dass die genaue Dokumentation der Fehler wichtig ist. Zum einen werden Tester/innen, je länger das Projekt dauert zu Applikations-Experten. Er kann oft relativ genau bestimmen, wo die Ursache des Fehlers liegt. Dies erspart dem Entwickler viel Zeit in der Ursachen-Analyse. Zudem ist es wichtig den Fehler so ausführlich zu dokumentieren, dass dieser auch Wochen später behoben werden kann. Nicht jeder Fehler hat eine gleich hohe Auswirkung auf den Betrieb und ist entsprechend zu priorisieren.

2.4 Sprint 0

Der Sprint 0 wird oft in den ersten Wochen des Projektes gemacht und ist eine Sonderform in der agilen Entwicklung. Er eignet sich hervorragend für grundlegende Vorbereitungsarbeiten in einem Projekt. Heuer (2014) beschreibt, dass in der ersten Woche des Projektes die Basis Infrastruktur aufgebaut wird (S. 17). Konkret könnte dies in einem ERP-Projekt die Aufsetzung der Entwicklungs- und Testumgebung sein sowie die Basis

Installation von Software. Auch sind erfahrungsgemäss Berechtigungen und Zugriffe der Teammitglieder ein Punkt, welche die Arbeiten unnötig verzögern, sofern diese nicht vorhanden sind. Speziell sind hier die Zugriffe von externen Entwicklern und Beratern zu erstellen, da diese aus Sicherheitsgründen oft zusätzliche Richtlinien einhalten müssen.

Im Kontext zum Testing kann im Sprint 0 die generelle Handhabung und Dokumentation der Tests definiert und ausgearbeitet werden. Aufgrund der den in den vorherigen Kapiteln erwähnet könnten dies folgende Themen beinhalten:

- Wie werden Anforderungen und Testfälle dokumentiert?
- Welche Teammitglieder sind verantwortlich für die Durchführung und Überwachung der Tests?
- Wie werden die Fehler nachverfolgt und behoben?
- Welche Kriterien müssen Testfälle erfüllen, damit sie als Regressionstest wiederverwendet werden können? Wie werden diese definiert und Rückverfolgt?
- Welche Werkzeuge werden für die Testfallverwaltung verwendet?
- Sind Mitarbeiter in der Anwendung dieser Werkzeuge geschult und haben sämtliche Mitarbeiter Zugriff auf diese Systeme?

Im Falle, dass mehrere Scrum-Teams an demselben Projekt arbeiten, sind die oben erwähnten Aufgaben umso wichtiger, sofern das Unternehmen ein einheitliches Vorgehen bezüglich Dokumentationen fordert. Falls bereits Werkzeuge für die Testdokumentation vorhanden sind und die Teams bereits Erfahrung mit Testing gemacht haben, sollten die oben erwähnten Punkte relativ schnell erledigt sein. Wenn jedoch noch Software beschafft werden muss, ist die Dauer von 2-4 Wochen (Sprint) erfahrungsgemäss eher knapp. So müsste vorgängig bereits ein geeignetes Werkzeug für die Testdokumentation evaluiert und installiert werden.

Erfahrungsgemäss ist der Sprint 0 bei einigen agilen Organisationen etwas verpönt (Scrum Academy, 2020). Grundsätzlich können viele der oben erwähnten Aufgaben auch in normalen Sprints erledigt werden. Einige Vorbereitende Aufgaben sind jedoch immer notwendig. Ob man diese nun in einem Sprint 0 erledigt, oder einfach als vorbereitende Aufgaben oder sogar als Vorprojekt deklariert, ist vermutlich zweitrangig. Wichtig ist, dass die Teams eine optimale Grundalge haben, effizient zu arbeiten.

3 ERP-Projekte

In diesem Kapitel wird auf die Eigenheiten von ERP-Projekten und ERP-Systemen eingegangen. Es wird untersucht, wie Teams zusammengestellt werden könnten, welche Unterschiede zu klassischer Softwareentwicklung bestehen, sowie weitere organisatorische Rahmenbedingungen. Zudem wird auch das Thema Schnittstellen und Ansätze zur Reduktion von Komplexität im Zusammenhang mit Schnittstellen beschrieben.

3.1 Agile Teams in ERP Projekten

Um agile Teams ein einem ERP-Projekt aufzubauen, sind unterschiedlicher Überlegungen notwendig. Im Normalfall werden in klassischen ERP-Projekten die Teilprojekte nach funktionalen Kriterien gegliedert und in spezifische ERP-Module eingeteilt. Bei funktionsorientierten Unternehmen, wie sie heute noch häufig anzutreffen sind, machte das absolut Sinn. So konnten in den jeweiligen Unternehmensabteilungen die Anforderungen ermittelt und entsprechenden mit Modul-Spezialisten umgesetzt werden. Robson (2016) beschreibt, dass auch agile Projekte grundsätzlich so gegliedert werden können. So können in grösseren ERP-Projekten mehrere agile Teams funktional nach Modulen gegliedert werden um die jeweiligen Module parametrisieren. Entwickler können in einem separaten Team gegliedert werden und, sofern die Anforderungen nicht durch Standard-Funktionalitäten abgedeckt werden, punktuell als Unterstützung hinzugezogen werden (S. 521). Diese Gliederung nach Funktionalen Kriterien hat jedoch wesentliche Nachteile. Es ist schwierig den Zusammenhalt und Kommunikation zwischen den Teams aufrecht zu erhalten. Robson (2016) beschreibt, dass eine solche Gliederung schnell zu einer «Wir gegen Sie» Mentalität führen kann und oft ein Kampf um Entwicklungsressourcen stattfindet (S. 528). Genau dieses Verhalten konnte in der Praxis auch festgestellt werden. So waren beim Betrieb der bestehenden ERP-Software die Aufgaben jeweils Funktional gegliedert. Jedes Team hatte ein grosses Interesse daran, die Funktionalitäten genau so zu gestalten, dass sie für die eigenen Bedürfnisse und Anspruchsgruppen optimal ausgerichtet waren. Zudem war der Kampf um Entwicklerressourcen jeweils analog der oben erwähnten Beschreibung. Dies führte oft zu aufwändigen und unnötigen Diskussionen. Der Fokus zum jeweiligen Ziel ging schnell verloren und es wurden unnötig Energie verschwendet.

In agilen ERP Projekten kann das Konzept der Modulverantwortlichen mit so genannten Feature-Teams (nicht zu verwechseln mit Feature-Test-Team) aufgenommen werden, was auch grundsätzlich fachlich sinnvoll ist. Entwickler müssen Bestandteil dieses Teams sein. Im Idealfall ist jedem Team ein fixer Entwickler zugeteilt und sofern die Entwicklungskapazitäten nicht ausreichend sind, notfalls Teilzeit (Robson, 2016, S. 535). Diese Teamzusammenstellung verbessert sicherlich die Kommunikation mit Entwicklungsressourcen im Team. Wenn diese jedoch knapp verfügbar sind, bleibt am Ende doch ein Engpass bestehen.

Was damit nicht gelöst wurde, ist das Problem der funktionalen Trennung. In der Literatur ist nicht abschliessend geklärt, wie in einem agilen ERP-Projekt die Features gegliedert werden sollen. Es besteht die Gefahr, dass Features stark nach funktionalen Kriterien

gegliedert werden, und so der Gesamtprozess und die Integration von Umsystemen aus dem Fokus gerät. Oft sind in ERP-Systemen Module in beispielsweise Materialwirtschaft, Beschaffung, Fertigung und Finanzen (Aufzählung nicht abschliessend) gegliedert. Mutmasslich stammt diese Modulaufteilung noch aus der Zeit der funktionalen Gliederung von Unternehmen. In modernen, prozessorientierten Unternehmen sollte jedoch ein grösseres Augenmerkt auf die Prozesse gelegt werden. Entsprechend sollte in Betracht gezogen werden, bei der Gliederung der Stories in Features darauf zu achten, dass diese prozessorientiert erfolgt und nicht funktionsorientiert. Klassisch in diesem Kontext ist der Wertefluss in einem ERP-System. Wenn beispielsweise Fertigung, Finanzen und Beschaffung von unterschiedlichen Feature-Teams entwickelt werden, kann die Definition des Werteflusses schwierig werden. So entstehen wieder funktionale Schnittstellen, welche dann oft mit «wir gegen sie» enden können.

Ein weiterer wichtiger Faktor bei der Definition der Features und dessen Zuteilung zu Scrum-Teams sind Schnittstellen. In ERP-Projekten sind Schnittstellen mitunter die grössten Risikofaktoren und oft sehr aufwändig zu erstellen. Zudem müssen in den meisten Fällen zusätzliche Ressourcen von Externen Systemlieferanten hinzugezogen werden, was in der agilen Entwicklung die Zusammenarbeit eher schwierig macht (Meier, 2020). So ist darauf zu achten, dass nicht mehrere Scrum-Teams an derselben Schnittstelle arbeiten. Mit diesem Vorgehen können auch Integrationstests einfacher durchgeführt werden da bei Anpassungen einer Schnittstelle lediglich die Stories eines Teams betroffen sind und entsprechend die notwendigen Testaktivitäten und Korrekturen einfacher zu handhaben sind.

Aufgrund der Erwähnungen in der Literatur sowie der Praxiserfahrungen kann hier sicherlich empfohlen werden, die Teams so zu gestalten, dass sie möglichst autonom arbeiten können und entsprechend sämtliche notwendigen Ressourcen direkt im eigenen Team verfügbar haben. Damit die Teams möglichst autonom arbeiten können, ist die Zusammenstellung der Features genau zu überlegen. Die genaue Zusammenstellung kann hier nicht klar bestimmt werden, zumal auch bei jedem Unternehmen individuelle Prozesse existieren. Aufgrund der modernen Organisationslehre ist der Prozessorientierte Ansatz bei der Definition und Zusammenstellung der Teams sicherlich zu berücksichtigen (Schmelzer & Sesselmann, 2013, S. 207).

3.2 Unterschiede zu klassischer Software Entwicklung

Ein wesentlicher Unterschied zwischen der Einführung eines ERP-System und der Entwicklung von Software ist, dass bei ERP-Einführungsprojekten vergleichsweise sehr wenig Code geschrieben wird. Hier ist klar die Seite des ERP-Anbieters und diejenige des Kunden, bei welchem die Software implementiert wird, zu Unterscheiden. Weiter ist es wichtig zu erwähnen, dass die Entwicklungszyklen der ERP-Systemanbieter sowie diejenigen der Kunden im Normalfall unabhängig voneinander stattfinden. Diese Tatsache ist relevant, sobald das ERP-System installiert und für den produktiven Betrieb freigegeben wurde. Dass sowohl der ERP-Systemanbieter als auch die Kunden das System in agilen Teams weiterentwickeln ist sicherlich realistisch. Jedoch findet üblicherweise zwischen diesen beiden Unternehmen keine detaillierte Abstimmung statt, wie es in agilen Organisationen wünschenswert wäre. Eine Unterscheidung zwischen genereller Entwicklung der Software (Releasemanagement) und Kundenspezifischen Anpassungen (Changemanagement) ist hier sicherlich hilfreich.

3.2.1 Release

Die generelle Entwicklung des ERP-Systems liegt komplett in Verantwortung des ERP-Systemanbieters. Der Kunde hat hier, vor allem bei grösseren Systemanbietern, wenig Einfluss auf den Produkt-Backlog. Den Kunden werden lediglich neue Releases zur Verfügung gestellt. Diese Änderungen betreffen oft den Kern der Software und sind (im weitesten Sinne) für sämtliche Kunden relevant.

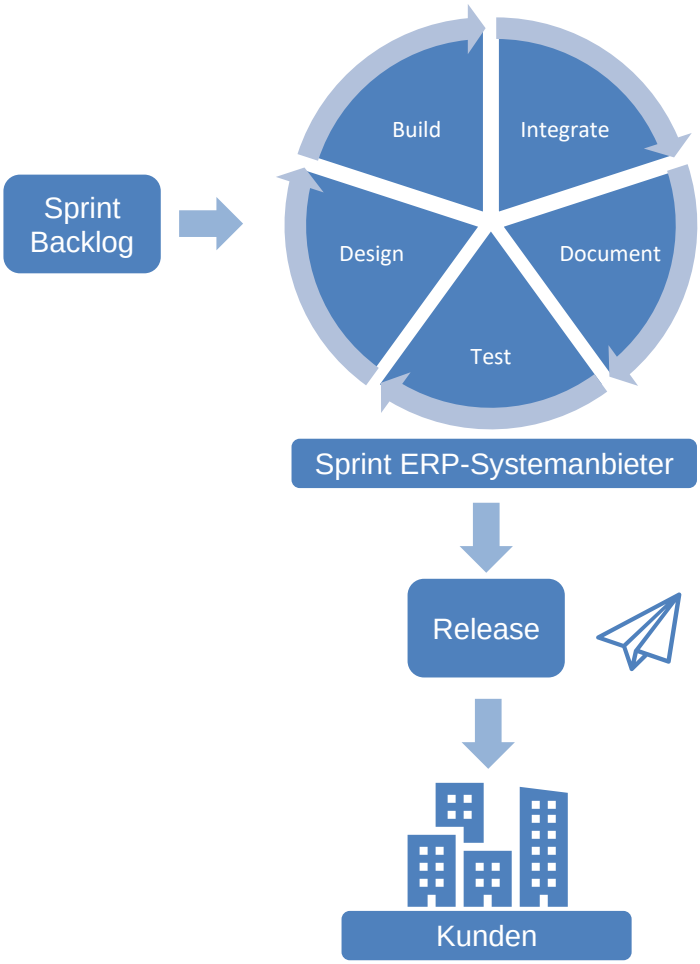

Abbildung 22: Generelle Entwicklung ERP-System durch ERP-Systemanbieter

Quelle: Eigene Darstellung

Im Bereich Testing ist hier die Herausforderung, dass der Kunde die Tests nicht nach einer klassisch agilen Vorgehensweise durchführen kann, da weder Stories noch Akzeptanztest in der agilen Organisation des Kunden definiert wurden. So ist der Kunde darauf angewiesen, dass das System von Seite des ERP-Systemanbieters bereits ausgiebig getestet wurde. Je nach Grösse des Releases müssen jedoch die wichtigsten Funktionalitäten auf Seite des Kunden ebenfalls getestet und verifiziert werden, bevor der Release in der Produktiv-Umgebung des Kunden freigegeben werden kann. Von der Ausprägung her wird der Kunde hier hauptsächlich Integrations- oder Systemtests durchführen, also grundsätzlich Testfälle des Feature-Test-Teams. Speziell zu Testen sind Individualanpassungen der Software. Diese sind erfahrungsgemäss nicht immer mit den Releases kompatibel und müssen etwas detaillierter getestet werden. Im Idealfall kann der Kunde hier auf die Testfälle zurückgreifen, welche im Rahmen der ERP-Einführung erstellt wurden (Hansmann, 2020). Voraussetzung ist jedoch, dass Integrations- und Systemtests durch die Teams dokumentiert und der Betriebsorganisation übergeben wurden. Diese Testfälle können dann laufend ausgeführt und notfalls angepasst werden, sofern sich funktional in der Software etwas geändert hat.

3.2.2 Changemanagement

Kundenspezifische Anpassungen liegen im Normalfall in der Verantwortung der Kunden. Hier werden oft Experten der Systemanbieter bei den agilen Teams der Kunden hinzugezogen. Die Change Requests werden vom Product Owner gesammelt, priorisiert und in einem oder mehreren Sprints umgesetzt. Somit eigentlich klassische agile Entwicklung.

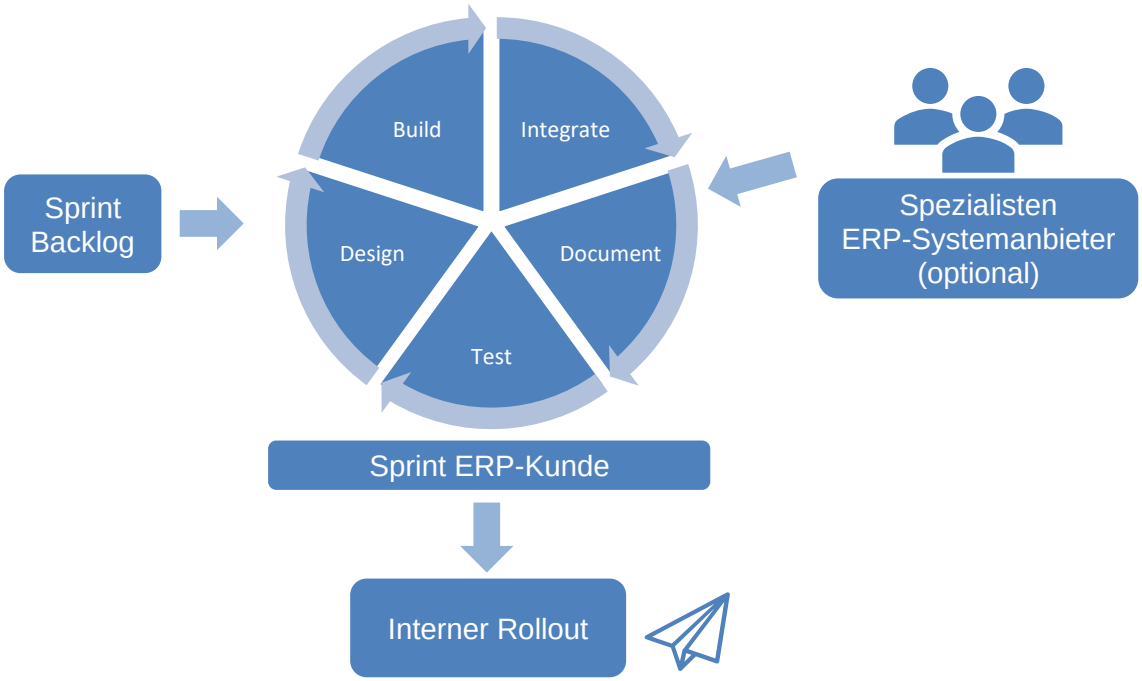

Abbildung 23: Individualentwicklung durch Kunde

Quelle: Eigene Darstellung

Im Bereich Testing kann der Kunde an dieser Stelle auf das klassische agile Vorgehen setzten. Jedoch ist es auch hier wertvoll, wenn das Scrum-Team oder das Testteam (je nach Organisation) bereits auf erstellte Testfälle zurückgreifen kann. Denn im Changemanagement ist der Fokus meist nicht mehr auf der Gesamtapplikation, wie es im Einführungsprojekt der Fall ist. So können bestehende Testfälle dabei helfen, die möglichen Auswirkungen des Changes besser zu beurteilen. Sofern neue Testfälle entstehen ist zu überlegen, ob diese auch für die Betriebsorganisation und das Releasemanagement relevant sind. Entsprechend ist die Dokumentation der Tests sinnvollerweise nachzuführen, analog der agilen ERP-Einführung.

3.2.3 ERP-Entwicklung

ERP-Systemanbieter betreiben Software-Entwicklung als Kernkompetenz. In einer «perfekten ERP-Welt» würden alle Weiterentwicklungen der Software «als Software-Standard definiert» und via Releases sämtlichen Kunden zur Verfügung gestellt. Dies hätte für ERP-Anbieter den Vorteil, dass genau eine Softwareversionen bei sämtlichen

Kunden im Einsatz steht, was den Support sowie die Entwicklung und das Testing wesentlich vereinfacht. Einige ERP-Anbieter versuchen konsequent diesen Weg zu gehen. Auf dem Markt sind diesbezüglich auch einige Ansätze zu erkennen, beispielsweise ERP-Systeme die nur noch als SaaS bezogen werden können und auf den Servern des ERP-Anbieters betrieben werden.

Bei dieser Art von Weiterentwicklung hat der Kunde nur am Rande Berührungspunkte mit dem Testing. Grundsätzlich ist hier der ERP-Kunde lediglich ein Stakeholder und liefert seine Stories in den Backlog des ERP-Anbieters. Die Sprints übernimmt der ERP-Anbieter vollständig und ist auch dafür verantwortlich, dass die Software ausführlich getestet und sämtlichen Kunden in Form von Releases ausgeliefert wird (siehe Abbildung 23). In dieser «perfekten ERP-Welt» sind diese Releases mit früheren Einstellungen und Setups kompatibel und dem Kunden stehen über Nacht, beinahe unbemerkt, zusätzliche Funktionalitäten zur Verfügung.

Dass die Realität nicht immer so perfekt ist zeigte sich in Vergangenheit bei einigen Systemanbietern, welche in grossen Releasen sehr viele Altlasten beseitigten und die Kunden für den Wechsel auf die neuste Version Kosten im Rahmen einer Neueinführung tragen mussten.

3.2.4 ERP Parametrisierung

Auch wenn die «perfekte ERP-Welt» für viele Unternehmen eine Utopie ist, verweilen wir in diesem Absatz noch kurz darin. In einer perfekten ERP-Welt wird dem Kunden die Software installiert (ob SaaS oder auf einem eigenen Server spielt eine untergeordnete Rolle) und dieser kann die Software anschliessend auf seine Bedürfnisse parametrisieren. Robson (2016) beschreibt, das dies im Normalfall durch Konfiguration-Spezialisten von Seite des ERP-Anbieters umgesetzt werde (S. 521). Dies ist ein wesentlicher Unterschied zur Software-Entwicklung. Eine Parametrisierung der Software verändert den programmierten Code der Software nicht. Konkret kann der Kunde zwischen unterschiedlichen Parametern denjenigen wählen, welche für seine Unternehmensprozesse am besten anwendbar sind. So bleibt die Software im entwickelten Standard und viele Tests müssen vom Kunden nicht selber durchgeführt werden. Beispielsweise kann der Kunde parametrisieren, ob auf seinen Rechnungen Zwischensummen je Seite gebildet werden sollen. Er sollte zwar testen, ob die Summen anschliessend auf den Belegen angezeigt werden, jedoch muss er nicht prüfen ob die Summen korrekt berechnet werden, da diese Tests in einer «perfekten ERP Welt» bereits vom ERP-Anbieter in der Entwicklung gemacht wurden.

Diese Tatsache ist sehr wichtig und zentral für die Auswahl und Definition der Tests bei einer Einführung von ERP-Systemen. Wo teamunterstützende und technikorientierte Tests bei der Entwicklung eines ERP-Systems zentral sind, sind bei der Einführung von ERP-Systemen die fachlich orientierten Tests umso wichtiger. Denn wenn die Entwickler/innen der Software korrekt gearbeitet und getestet haben, und die Software vom Kunden lediglich parametrisiert wird, sind fachliche Tests aus Kundensicht wichtiger als technische. Spannenderweise lässt sich aus dieser Tatsache auch den Schluss ziehen, dass Testautomatisierungen bei einer ERP-Einführung wesentlich weniger Potential hat als

auf Seite der ERP-Anbieters, welcher den Code der Software verändert und entsprechend viel mehr technisch orientierte Test ausführen muss.

3.2.5 ERP-Individualanpassungen

Nun verlassen wir die «perfekte ERP-Welt» wieder und begeben uns in die Realität des Unternehmertums. Bei rund 40% aller Grossunternehmen beträgt die Nutzungsdauer der eingesetzten ERP-Systeme gemäss einer Umfrage mehr als 10 Jahre. Bei rund 75% aller Unternehmen ist die Nutzungsdauer grösser als 5 Jahre (Becker & Ortmann, 2019, S. 7). Während dieser langen Nutzungsdauer werden oft Anpassungen an den Systemen vorgenommen. Dies nicht nur durch Veränderung der Systemparameter. Damit ein Unternehmen möglichst effizient arbeiten kann und auch Branchen- oder Unternehmensspezifische Prozesse korrekt abgebildet werden, sind Codeanpassungen an ERP-Systemen oft die Regel und nicht die Ausnahme. Sofern der ERP-Systemanbieter nicht bereit ist, diese Systemanpassungen als «Standard» zu entwickeln, ist der Kunde im Bereich Testing mehr gefordert als sonst. Software-Entwickler/innen stehen oft in engem Kontakt mit dem Kunden und erledigt diese Tätigkeiten nicht in regulären Sprints auf Seite des Systemanbieters. Sofern der Kunde seinerseits eine agile Organisation besitzt, werden Entwickler/innen oft vorübergehend in das agile Team des Kunden aufgenommen. So verlagert sich einen grossen Teil der Testaktivitäten vom Systemanbieter in die agile Organisation des ERP-Kunden (siehe Abbildung 24: Individualentwicklung durch Kunde).

Zusätzlich zu den individuellen Softwareanpassungen haben Schnittstellen zu Umsystemen ebenfalls einen wesentlichen Einfluss auf das interne Testing. Grössere Unternehmen haben im Normalfall viele Schnittstellen zu anderen Applikationen. Diese sind meist so spezifisch, dass sie für jeden ERP-Kunden individuell umgesetzt werden müssen. Zwar wurden in Vergangenheit wesentliche Fortschritte in den Schnittstellen Technologien gemacht, jedoch können bei Weitem nicht alle Systeme über «Standard Technologien» angesprochen werden. Des Weiteren hat der ERP-Systemanbieter keinen Einfluss auf die Funktionalität der anzubindenden Applikationen. Als Schlussfolgerung daraus muss auch hier die Entwicklung dieser Schnittstellen durch die agile Organisation des ERP-Kunden erfolgen und entsprechend ist auch das Testing technisch orientierter als wenn das ERP-System lediglich parametrisiert werden kann.

3.3 Organisation

Die organisatorischen Rahmenbedingungen sind im Referenzprojekt eine grosse Herausforderung. Einerseits wird das Projekt mit unterschiedlichen externen Unternehmen umgesetzt, andererseits auch mit internen Ressourcen aus den unterschiedlichen Fachbereichen. Das Projekt geniesst bei der Geschäftsleitung eine sehr hohe Priorität und wird als Schlüsselprojekt bezeichnet. Aus diesem Grund wurden mehrere Stellen geschaffen, welche sich ausschliesslich mit diesem Projekt auseinandersetzten. Mit der Bildung eines Kernteams, bestehend aus Projektleiter, Prozess- und Fachspezialisten, wurde eine Basis für dieses Projekt geschaffen. Weitere Fachspezialisten aus den

Geschäftsbereichen werden situativ hinzugezogen. Die Teams für die Umsetzungsphase respektive der Sprints wurden bis anhin noch nicht definiert. Mitglieder des Kernteams sind jedoch mit Sicherheit auch in diesen Teams vertreten.

3.4 Testen in ERP Projekten

ERP-Projekte wurden in der Vergangenheit hauptsächlich nach dem Wasserfall-Modell umgesetzt. Dadurch wurden zuerst die Anforderungen erstellt, das System konfiguriert und anschliessend die Testfälle erstellt und ausgeführt. Dabei wurden die Tests entlang der Prozesse ausgeführt, welche grundsätzlich Integrationscharakter hatten. Diese Vorgehensweise hatte sich über Jahrzehnte etabliert und ist stark in den Köpfen der Anwender und Projektmitglieder verankert. In agilen ERP-Projekten muss dies genau in umgekehrter Reihenfolge stattfinden. Die Akzeptanzkriterien sowie die dazugehörigen Tests sollten erstellt sein, bevor der Prozess im System konfiguriert wird (Robson, 2016, S. 1386). Generell sind erfolgreiche Tests in einem agilen Projekt eine Voraussetzung dafür, dass die Story abgeschlossen werden kann (Definition of Done). Die Herausforderung hierbei ist jedoch, wie mit einzelnen Stories und Testfällen Integrationstest durchgeführt werden können. Robson (2016) beschreibt hier, dass im Grundsatz einfach Tests von einzelnen Stories genommen werden können, und damit ein grösseres Testszenario zu entwickeln (S. 2378). So entstehen Testfälle, welche in einem ähnlichen Masse prozessorientiert ablaufen und der Anwender sich sehr gut damit identifizieren kann.

3.5 Erfolgsfaktoren und Risiken

Einer der grössten Erfolgsfaktoren in einem solchen agilen Projekt ist die Verfügbarkeit der Fachspezialisten. Diese sind oft auch in anderen Projekten und Vorhaben involviert und entsprechend schwer verfügbar. Das starke Commitment der Geschäftsleitung ist sehr positiv zu werten. So besteht die Möglichkeit, Prioritäten neu zu setzten und dadurch notwendige Ressourcen zu Gunsten dieses Vorhabens zu entlasten.

Als grosses Risiko wird die hohe Integration des ERP-Systems in Umsysteme betrachtet. So sind im Verlaufe der Umsetzung diverse Schnittstellen neu zu erstellen und bestehende abzulösen. Entsprechend wird die Verfügbarkeit von Schlüsselpersonen wie beispielsweise internen Schnittstellenexperten und auch Experten von Umsystemen entscheidend sein für eine schnelle und reibungslose Umsetzung. Diese Ressourcen sind jedoch nur sehr beschränkt vorhanden. Eine agile Einführung des Systems inklusive produktiv-Setzung der Releases wird sich voraussichtlich als sehr schwierig gestalten. So ist bei der Definition von Features und Releases ein spezielles Augenmerk zu setzten. Gelingt es dem Unternehmen nicht, die Features und Releases so zu planen, dass eine schrittweise Einführung möglich ist, muss das System per Stichtag im gesamten Unternehmen ausgerollt werden. Dieser «Big Bang» ist bei Unternehmen dieser Grösse sowohl organisatorisch als auch technisch sehr anspruchsvoll. So müssen alle Mitarbeiter gleichzeitig geschult werden und sämtliche Schnittstellen werden am Tag X umgestellt. Treten hier gleichzeitig mehrere Fehler / Probleme auf, ist es schwierig, diese zeitnah zu korrigieren. Auch ein Rollback ist ab einem gewissen Zeitpunkt nahezu unmöglich. So ist eine schrittweise Einführung mit gut getesteten Features / Releases, wie es

agile Vorgehensweisen proklamieren, mit Sicherheit ein Faktor, der entscheidend sein kann für den Erfolg.

3.6 Systemintegration/Schnittstellen

In einer «perfekten ERP Welt» werden alle im Unternehmen benötigten Funktionalitäten durch das ERP-System zur Verfügung gestellt. Konkret würde das bedeuten, dass das Unternehmen genau eine Applikation auf ihrem Server installiert hat; das ERP-System. Die Realität sieht jedoch ganz anders aus. Spezialsoftware wird in den Unternehmen zu hundertfach eingesetzt. Je grösser das Unternehmen ist, desto mehr Applikationen sind im Einsatz. Als klassisches Beispiel können wir hier die Personalstammdaten verwenden. Diese Daten werden in nahezu allen Applikationen benötigt:

- ERP-System
- Zeit- und Leistungserfassung
- Zutrittssysteme
- Windows Benutzeraccount
- Softwareverteilung
- HR-, Lohn- und Recruiting Systeme
- Benutzeraccounts in sämtlichen Applikationen
- … usw.

Im Zeitalter der Digitalisierung werden diese Systeme je länger je mehr miteinander vernetzt. Zum einen ist so die Qualität sowie die Durchgängigkeit der Daten sichergestellt, zum anderen ist die Mehrfacherfassung von Stammdaten sehr aufwändig, verlangsamt den Prozess und muss entsprechend automatisiert werden. So werden die Personalstammdaten im ERP-System zentral erfasst und mit Hilfe von Schnittstellen an Umsysteme verteilt. Keine Angst: Auf eine technische Abhandlung von Schnittstellentechnologien wird hier verzichtet. Was jedoch im Kontext dieser Arbeit hervorgehoben werden muss, ist dass das ERP-System als Kernapplikation im Unternehmen mit nahezu allen Umsystemen kommunizieren muss. So befasst sich ein wesentlicher Teil der Entwicklungsarbeit bei einer ERP-Einführung mit der Integration und Anbindung von Umsystemen.

In der Entwicklung und im Testing sind Schnittstellen eine besondere Herausforderung. Es braucht unterschiedliche Experten, welche bei der Entwicklung der Schnittstelle eng zusammenarbeiten (Meier, 2020). Auf der einen Seite ist der ERP-Spezialist, welcher die Datenstruktur im System kennt und auch die zu übermittelnden Daten definieren und interpretieren kann. Dazu kommt der Schnittstellen-Experte, welcher die Kommunikation zwischen den beiden Systemen entwickeln muss. Als weitere Person wird ein Experte des Zielsystems benötigt, welcher die Übermittlung der Daten spezifizieren und deren korrekte Übermittlung verifizieren muss.

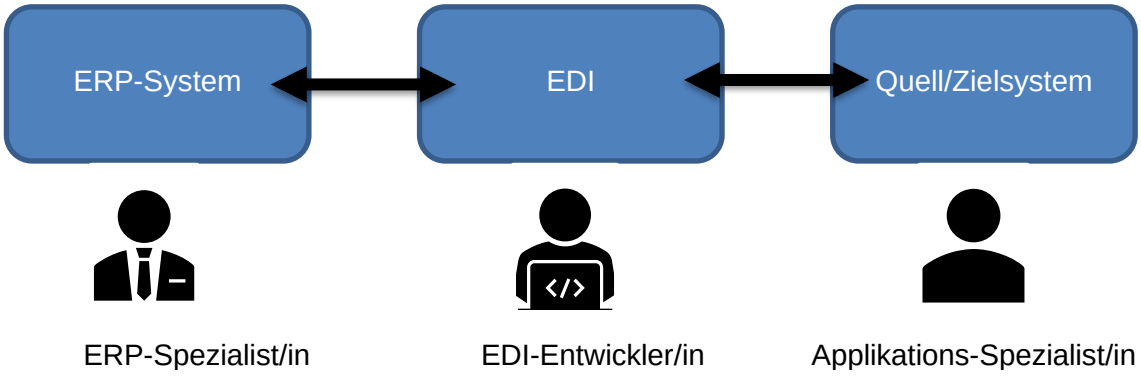

Abbildung 24: Entwicklung von Schnittstellen

Quelle: Eigene Darstellung

Aus organisatorischer Sicht werden sowohl ERP- als auch EDI-Spezialist/innen mutmasslich über die gesamte Projektdauer Bestandteil des Scrum-Teams sein. Die Applikationsspezialist/innen wiederum werden im Team lediglich für die Dauer der Schnittstellenentwicklung benötigt. Obwohl Scrum-Teams im Normalfall relativ beständig bleiben, kann hier sicher empfohlen werden, den Applikationsspezialist/innen für die Umsetzung dieser Sprint fix in das Team aufzunehmen. So wird das Team befähigt, die Stories rund um die Anbindung der Spezialapplikation selbständig und schnell umzusetzen. Eine weitere und nicht zu unterschätzende Herausforderung sind die Tests der Schnittstelle. Auch hier können und müssen Applikationsspezialist/innen einen wesentlichen Beitrag zur Erstellung der Testfälle leisten. Denn nur sie können auf Seite der Spezial Applikation beurteilen, ob der Test wirklich erfolgreich ist und die Daten korrekt übermittelt werden. Dies ist ein weiterer Grund, den Applikationsspezialist/innen für eine gewisse Dauer fix in das Scrum-Team zu integrieren. Nur so kann im Sprint sofort getestet werden, was in der agilen Entwicklung vorausgesetzt wird.

4 Handlungsempfehlungen

Die nachfolgend erfassten Handlungsempfehlungen richten sich an mittelgrosse Organisationen, welche ein ERP-Einführungsprojekt agil abwickeln möchten. Im Fokus dieser Handlungsempfehlungen ist das Scrum Framework. Hier ist zu erwähnen, dass die Scrum-Teams eine grosse Eigenverantwortung für die Erstellung und das Testing des Produktes haben. So sind die nachfolgenden Empfehlungen nicht als statische Richtlinie, sondern mehr als Inspiration für die Abwicklung eines agilen ERP-Projektes mit Fokus auf Testing zu verstehen. Wichtig ist jedoch, dass sich das Projektteam bewusst mit dieser Thematik befasst und das für seine Organisation und dessen Rahmenbedingungen optimale Vorgehen definiert.

4.1 Personas im agilen ERP Projekt

Aufgrund der theoretischen Aspekte im agilen Testing sowie den Erfahrungen aus ERP-Projekten sind nachfolgend unterschiedliche Personas aufgelistet. Diese Personas zeigen auf, welche Fähigkeiten im Bereich Testing gefordert werden und unterstützen die Organisation entsprechend bei der Bildung der Teams. Zudem können diese Personas dazu dienen, die Aufgaben und Verantwortung in dem agilen ERP-Teams zu definieren. Natürlich müssen diese Rollen nicht zu 100% fixiert werden, das würde den Grundsätzen von Agilität zuwider sprechen. Denn: «Individuen und Interaktion stehen über Prozessen und Werkzeugen» (Gloger, 2016, S. 24). Jedoch ist eine Auseinandersetzung mit den Aufgaben im Bereich Testing innerhalb von Scrum-Teams empfehlenswert. Die Aufgeführten Personas müssen nicht zwingend als einzelne Personen betrachtet werden. Je nach Konstellation kann auch eine Person die Fähigkeiten von mehreren Personas abdecken.

4.1.1 Entwicklungs-Persona

Entwicklungs-Personas sind hauptsächlich bei Individualentwicklungen des ERP-Systems im Einsatz. Ihre Tests sind sehr technisch orientiert. Sie sind die ersten Personen, die Tests bei Individualentwicklungen ausführen. Der selbst erstellte Code wird mit den vorab definierten Testfällen laufend überprüft. Sofern Testautomatisierung eingesetzt wird, programmieren sie Testfälle, bevor der eigentliche Code entwickelt wird. Im Normalfall werden Sie von Seite des ERP-Systemanbieters gestellt oder sind unternehmensintern für Schnittstellen verantwortlich.

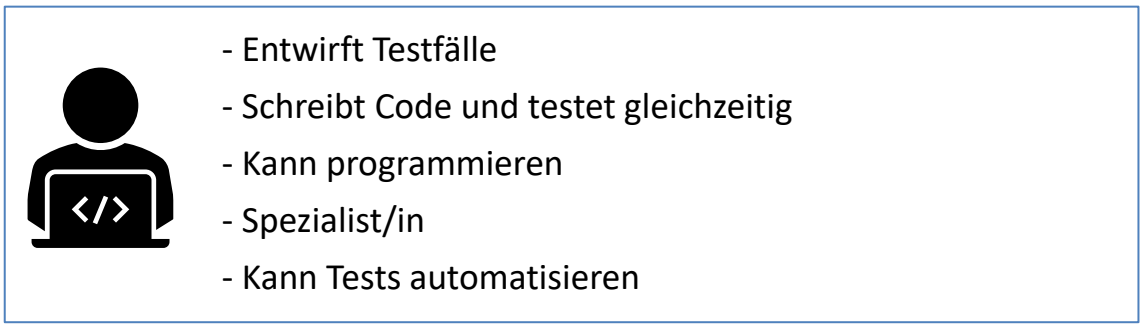

Abbildung 25: Entwicklungs-Persona

Quelle: Eigene Darstellung (Symbol aus Microsoft Word)

4.1.2 Test-Persona

Test-Personas haben den Gesamtüberblick über die Testaktivitäten. Sie arbeiten Vollzeit im Scrum-Team mit und unterstützen die Teammitglieder bei den Testaktivitäten. Wenn ein Feature-Test-Team existiert, übergeben sie ausgewählte und repräsentative Testfälle der Testmaster-Persona. Ansonsten begleiten sie die Testfälle in ihrem gesamten Lebenszyklus, von der Entstehung in den Sprint-Plannings bis zu den Integrations- und Systemtests. Bei Projektabschluss übergeben sie ausgewählte Testfälle an den operativen Betrieb. Mit Ihrer Unterstützung finden die Tests laufend und koordiniert statt. Durch eine angemessene Dokumentation stellt die Test-Persona sicher, dass der Stand der Tests zu jeder Zeit nachvollziehbar ist. Auch mit anderen Scrum-Teams findet ein regelmässiger Austausch statt, wo Test-Erfahrungen ausgetauscht werden und die Testdokumentation koordiniert wird.

Diese Persona ist technikaffin und kann gleichzeitig auch die Unternehmensprozesse gut. Durch Ihr vernetztes Denken versteht sie die Zusammenhänge der erstellten Applikation und kann nebst der Unterstützung der Entwickler/innen bei Unit-Tests ebenfalls bei den Integrations- und Systemtests aktiv mitarbeiten. In einem früheren Leben war diese Persona möglicherweise als Requirements Engineer oder Business Analyst tätig.

- Ist technikaffin
- Versteht Logik und Zusammenhänge von:
    - Applikationen
    - Funktionen und Prozesse
    - Datenstrukturen
- Erstellt und dokumentiert Testfälle
- Hat den Blick fürs Ganze
- Ist kommunikativ

Abbildung 26: Test Persona

Quelle: Eigene Darstellung (Symbol aus Microsoft Word)

4.1.3 Fachexperten-Persona

Diese Personas zeichnen sich darin aus, dass sie End to End Prozesse im Unternehmen detailliert kennen und verstehen. Viele der erstellten User Stories stammen aus ihren Federn. Sie kennen die Bedürfnisse und Anforderungen im Betrieb und unterstützen das Scrum-Team auch bei prozessualen Fragestellungen. Bei Integrations- und Systemtests können sie das Scrum-Team respektive das Feature-Test-Team wesentlich unterstützen. Ihre Expertise erlaubt es, den Release aus fachlicher Sicht für den operativen Betrieb frei zu geben. Auch wenn sie möglicherweise nicht Vollzeit im Team engagiert sind, nehmen sie im Normalfall an sämtlichen Teammeetings teil und kennen den Stand der Entwicklung gut. Sie sind KeyUser in ihren Organisationseinheiten.

- Kennt die Prozesse im Detail
- Kann Akzeptanztest formulieren
- Kennt die Bedürfnisse der Anwendet gut
- Kann das System bedienen
- Kann Tests nachvollziehen und beurteilen

Abbildung 27: Fachexperten-Persona

Quelle: Eigene Darstellung (Symbol aus Microsoft Word)

4.1.4 Systemkonfigurations-Persona

Systemkonfigurations-Personas kennen das ERP-System und dessen Funktionalitäten im Detail. Sie werden von Seite des ERP-Systemanbieters zur Verfügung gestellt. Gemeinsam mit den Fachexperten konfigurieren / parametrisieren sie das System. Aufgrund Ihrer Systemkenntnisse kennen sie die die Auswirkungen von Systemeinstellungen und unterstützen die Tester-Persona sowie die Fachexperten bei der Definition und Erstellung der Testfälle. Sie können zudem beurteilen, auf welche Tests ein besonderes Augenmerk gelegt werden muss.

- Konfiguriert die Software
- Unterstützt bei der Definition von Testfällen
- Ist ERP- Spezialist/in
- Kann Kritikalität und Auswirkung von Konfigurationen beurteilen

Abbildung 28: Systemkonfigurations-Persona

Quelle: Eigene Darstellung (Symbol aus Microsoft Word)

4.1.5 Anwendungs-Persona

Diese Persona ist nicht Mitglied eines Entwicklungs-Teams. Je nach Herausforderung kann Sie jedoch unterstützend hinzugezogen werden. Bei einzelnen Stories welche ihren Fachbereich betreffen, kann sie auch Akzeptanztests durchführen. Sie kennt ihre eigenen Prozesse gut, lernt schnell und ist geschickt in der Anwendung von Software. Angeleitet durch die Test-Persona, Testmaster-Persona, oder die Fachexperten-Persona kann sie zum Beispiel bei Explorativen Testsessions einen wertvollen Beitrag leisten. In Produktedemonstrationen wird ihr nach jedem Sprint der Projektstand mitgeteilt, so ist sie stets auf dem Laufenden. Sie ist interessiert und freut sich auf das neue ERP-System.

- Geschickt in der Anwendung von Softwase
- Kennt die eigenen Prozesse gut
- Ist experimentierfreudig und Interessiert

Abbildung 29: Anwendungs-Persona

Quelle: Eigene Darstellung (Symbol aus Microsoft Word)

4.1.6 Spezialapplikations-Persona

Für die Anbindung von Umsystemen an das ERP-System ist diese Persona unersetzlich. Sie ist verantwortlich für Systeme, welche in Form von Schnittstellen an das ERP-System angebunden werden. Sie kann die Ergebnisse der Schnittstellenübermittlung fachlich in ihrer Applikation prüfen. Zudem unterstützt sie die Entwicklungs-Persona bei der Spezifikation der Schnittstelle. Gemeinsam mit der Test- oder Testmaster-Persona entwickelt sie Testfälle für Integrations- und Systemtests im Kontext zum anzubindenden System. Diese Persona ist nicht dauerhaftes Mitglied des Scrum-Teams. Sie wird lediglich in denjenigen Sprints dazu genommen, in welchen die Schnittstelle zu ihrer Applikation erstellt wird. Zusätzlich nimmt sie bei Feature-Tests teil.

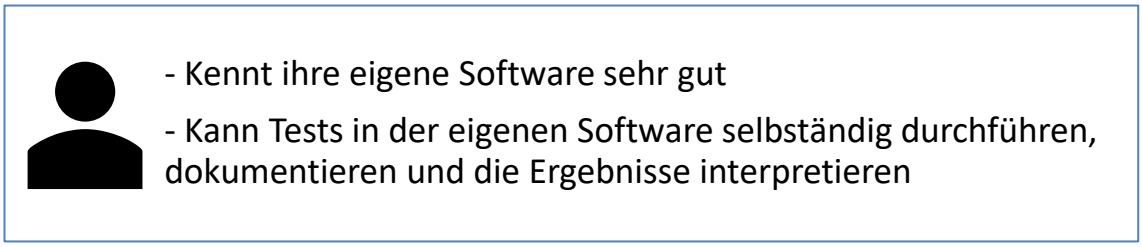

Abbildung 30: Spezialapplikations-Persona

Quelle: Eigene Darstellung (Symbol aus Microsoft Word)

4.1.7 Spezialtest-Persona

Diese Personas sind ebenfalls nicht Teil des Scrum-Teams und werden nur bei Spezialtests hinzugezogen. Sie sind beispielsweise Experten für Sicherheit oder können Last und Performance Tests durchführen. Sie sind in ihrem Gebiet absolute Spezialisten. Sie können sowohl von der eigenen IT-Abteilung als auch von externen Dienstleistern punktuell hinzugezogen werden. Sie werden nicht vom ERP-Systemanbieter gestellt und sind somit in ihrer Beurteilung neutral.

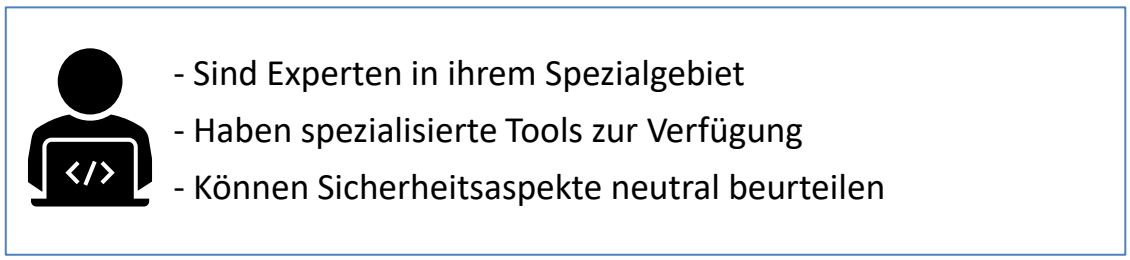

Abbildung 31: Spezialtest-Personas

Quelle: Eigene Darstellung (Symbol aus Microsoft Word)

4.1.8 Testmaster-Persona

Die Testmaster-Persona hat eine spezielle Rolle. In grossen ERP-Projekten koordiniert sie sämtliche Testaktivitäten ausserhalb der Entwicklungs-Teams. Sie steht sowohl in engem Austausch mit den Tester-Personas, als auch mit dem Product Owner. Zusammen mit ihrem Testteam führt sie strukturiert Feature und Systemtests durch, dokumentiert Fehler und empfiehlt die Freigabe der Releases. Durch die Dokumentation und Auswertung von Testaktivitäten wissen alle Beteiligten, wie es um den Entwicklungsstand sowie die Qualität der erstellten Software steht. Sie ist sehr kommunikativ und pflegt Beziehungen zwischen verschiedenen Anspruchsgruppen, von der Geschäftsleitung über Scrum-Teams bis zu den Anwendern. Sie kennt die Geschäftsprozesse gut und kann die Fachexperten-Personas und Anwendungs-Personas bei den Testaktivitäten zu Höchstleistungen bringen. Die Dokumentierten Testfälle übergibt sie nach Projektabschluss der Betriebsorganisation.

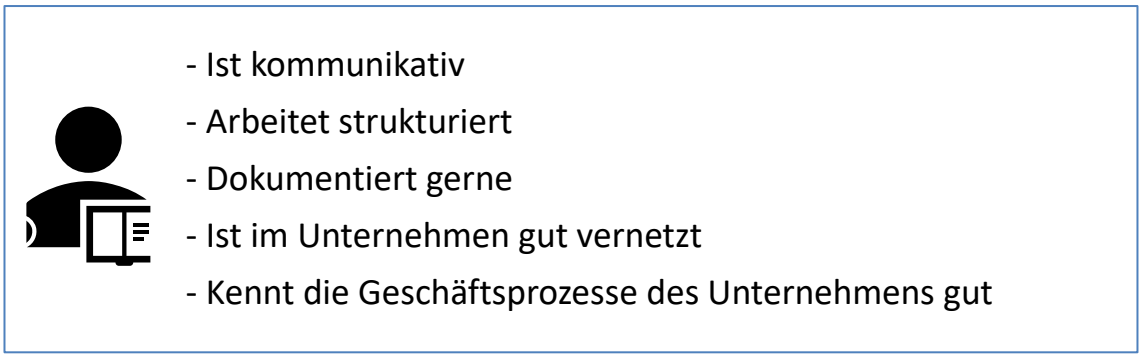

Abbildung 32: Testmaster-Personas

Quelle: Eigene Darstellung (Symbol aus Microsoft Word)

4.2 Organisatorische Einbettung

Die Organisation ist sowohl für die Entwicklung als auch für die Testaktivitäten entscheidend. Nachfolgend wird aufgezeigt wir die einzelnen Teams zusammengestellt werden können, zu welchem Zeitpunkt sie Testen und welche Vor- und Nachteile die gewählte Organisationsform hat.

4.2.1 Testaktivitäten innerhalb von Scrum-Teams

In dieser Zusammenstellung werden möglichst alle Testaktivitäten durch das Scrum-Team selbst durchgeführt. Prozessspezialisten und Key-User aus den Fachbereichen sind Bestandteil (Voll- oder Teilzeit) des Teams und können Systemtests sowie einen grossen Teil der Abnahmetests selbständig durchführen. Dadurch wird das Scrum-Team tendenziell grösser.

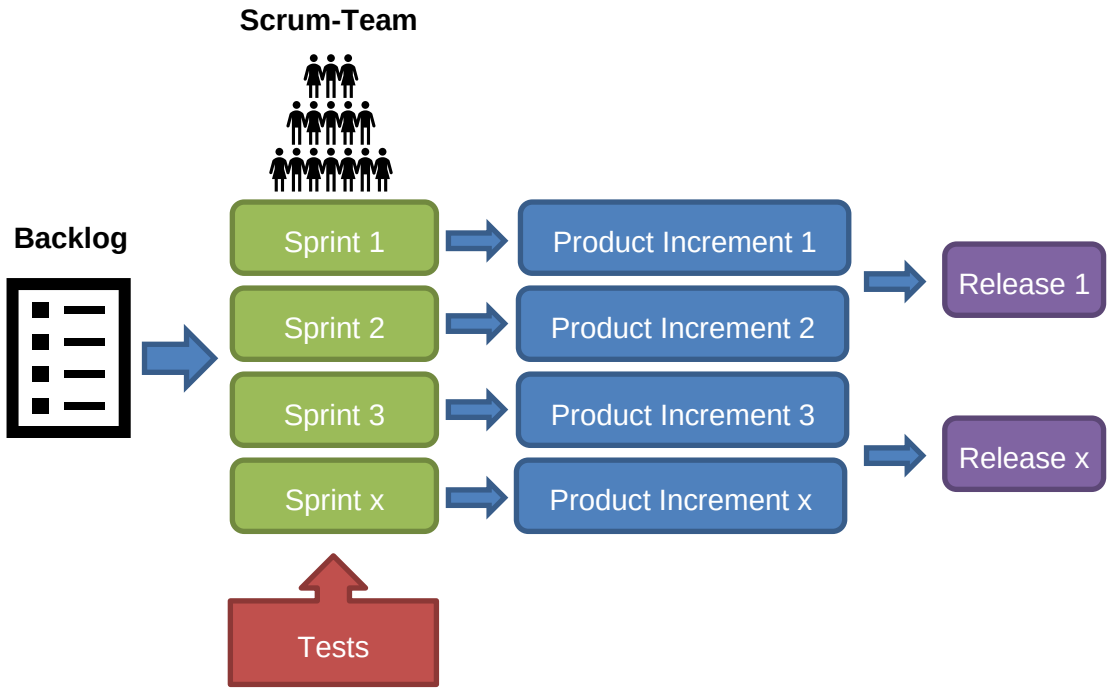

Abbildung 33: Alle Testaktivitäten finden innerhalb der Scrum-Teams statt

Quelle: Eigene Darstellung

Mit dieser Konstellation wird der Grundsatz gestärkt, dass «das Scrum-Team aus Personen bestehen muss, welche die Kenntnisse haben, das Produkt in ihrer Gesamtheit zu liefern» (Gloger, 2016, S. 92). Punktuell könnten weitere Stakeholder für Akzeptanztest hinzugezogen werden, falls in einigen Themengebieten das Fachwissen fehlt, oder in einem Themenbereich nur wenige Stories existieren.

Vorteile

Diese Organisationsform hat den wesentlichen Vorteil, dass die Kommunikationswege sehr kurz sind und die Tests unmittelbar stattfinden. Auch die Fehleridentifikation sowie die Ursachenanalyse ist effizienter, da sämtliche Tests innerhalb der Teams stattfinden. Zudem haben die zukünftigen Key-User und Fachexperten einen tieferen Einblick in die ERP-Prozesse und können mit diesem Knowhow den operativen Betrieb wesentlich unterstützen.

Nachteile

Bei grossen Vorhaben mit mehreren Scrum-Teams werden die Testaktivitäten je länger je umfangreicher. Mit der Erstellung und Durchführung von Integrations- und Systemtests muss innerhalb der Sprints in kurzer Zeit viel getestet werden. Zudem ist die Zusammenarbeit in grossen Teams schwieriger als in kleinen, eingespielten Teams. Sofern mehrere Scrum-Teams beteiligt sind, fehlen Personen oder Teams, welche die übergreifenden Tests koordinieren.

Empfehlung

Für mittelgrosse ERP-Projekt eher nicht geeignet. Der Testumfang würde die Sprints extrem verlangsamen, die Kommunikation wird schwierig und Testkoordination zwischen den Teams sehr komplex.

4.2.2 Autonomes Feature-Test-Team

Diese Organisationsform beinhaltet nebst den Entwicklungs-Teams ein separates und unabhängig organisiertes Feature-Test-Team. Dieses wird mit Fachexperten und Anwendervertreter aus unterschiedlichen Bereichen zusammengestellt. Als Koordinator zwischen den unterschiedlichen Teams und Anspruchsgruppen wird die Testmaster Persona eingesetzt. Diese plant und koordiniert sämtliche Testaktivitäten ausserhalb der Entwicklungs-Teams.

In den Entwicklungs-Teams sind lediglich diejenigen Personas vertreten, welche für die Erstellung des Produktes effektiv notwendig sind. Diese testen hauptsächlich auf der Ebene Unit. Integrations- und Systemtests werden durch das Feature-Test-Team durchgeführt und protokolliert.

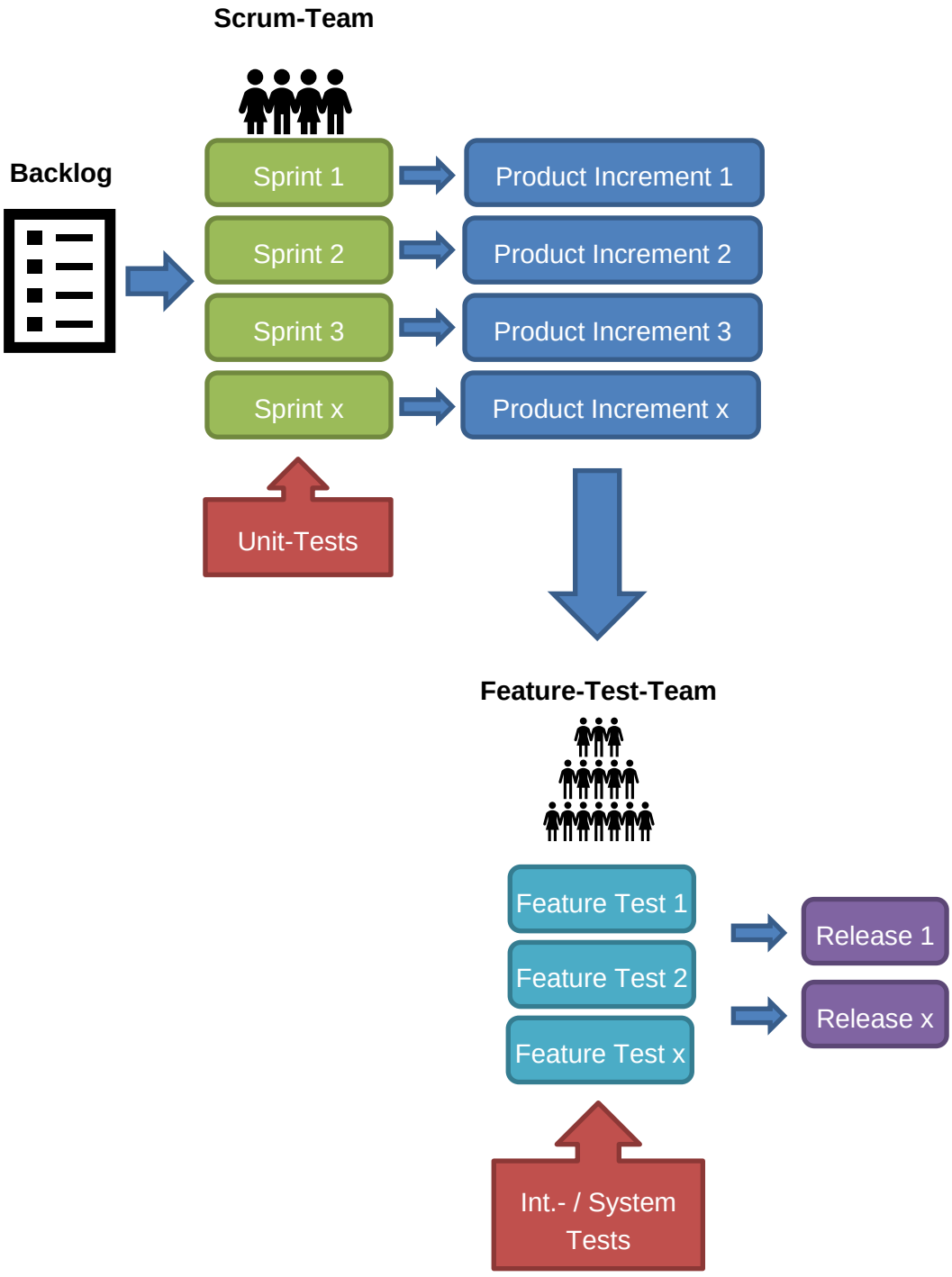

Abbildung 34: Autonomes Feature-Test-Team

Quelle: in Anlehnung an SwissQ Consulting AG (2020)

Vorteile

Mit der Erstellung von separaten Teams werden die Entwicklungs-Teams wesentlich entlastet. Das Feature-Test-Team kann sich stark auf die Integrations- und Systemtests fokussieren. Durch die Testmaster-Persona können die Tests strukturiert und auch über mehrere Features und Stories hinweg durchgeführt werden. Auch grosse Vorhaben

können mit dieser Organisationsform gut bewältigt werden. Die Weiterentwicklung wird durch die Testaktivitäten nicht unterbrochen.

Nachteile

Durch die komplette Separierung der Teams können Informationslücken entstehen, was beispielsweise die Fehleridentifikation erschwert. Auch die Zeitspanne zwischen Fehlererkennung und Problemidentifikation ist grösser. So besteht die Gefahr, dass mit fehlerhafter Software weiterentwickelt wird und daraus Folgefehler entstehen. Die Fehlerkosten sind entsprechend höher, als wenn die Tests direkt in den Teams durchgeführt werden.

Empfehlung

Für mittelgrosse ERP-Projekte geeignet. Die Grösse des Vorhabens bedingt auch auf der Test-Seite eine Skalierung. Mit dieser Organisationsform können auch zusammenhängende Stories und Features adäquat und ausgiebig getestet werden.

4.2.3 Test in Test-Sprints

Diese Organisationsform ist eine Kombination der beiden erstgenannten Varianten. Die regulären Sprints werden regelmässig durch Feature-Tests-Sprints unterbrochen. In diesen kommen Vertreter aus den unterschiedlichen Entwicklungs-Teams, Fachspezialisten und Anwendervertreter zusammen und testen die erstellten Stories und Features gemeinsam. Im Fokus dieser Sprints sind ebenfalls Integrations- und Systemtests. Koordiniert werden diese Sprints durch die Testmaster-Persona. Während der regulären Sprints kann sich diese Persona regelmässig mit den Test-Personas austauschen, Termine und Räumlichkeiten organisieren und Testfälle erstellen.

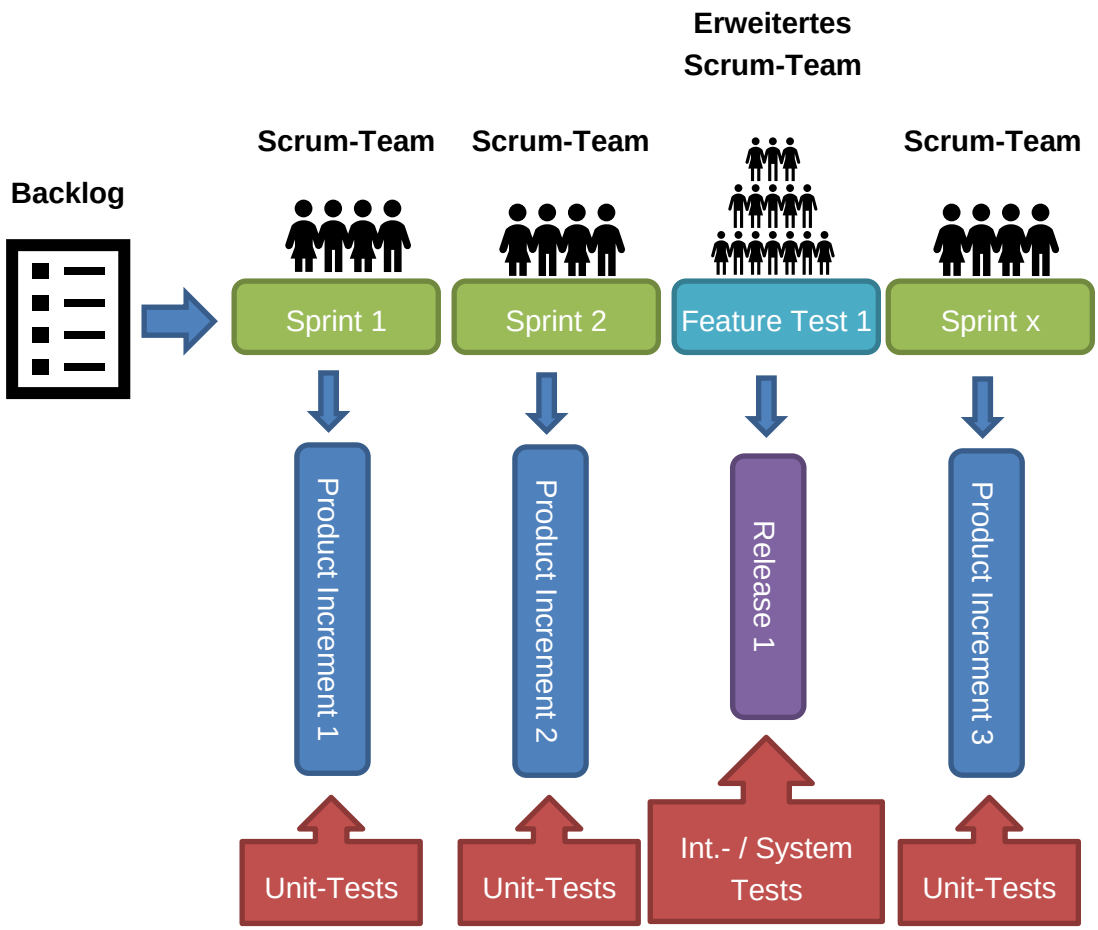

Abbildung 35: Testen in Feature-Test-Sprints

Quelle: Eigene Darstellung

Vorteile

Mit dieser Organisationsform werden einige Vorteile der genannten Varianten kombiniert und Nachteile eliminiert. Es werden ausführliche Integrations- und System-Tests durchgeführt. In den regulären Sprints kann sich das Entwicklungs-Team auf die Erstellung der Software fokussieren, aufwändige Integrationstests fallen dort weg. Durch die Zusammenkunft unterschiedlicher Teams und Stakeholder kann ein guter Austausch stattfinden. Die Fehler- und Problemidentifikation ist aufgrund der Anwesenheit von Entwicklungs-Mitgliedern effizienter. Die Kommunikationswege sind kürzer. Erkenntnisse aus diesen Test-Sprints können in die nachfolgenden Sprints sofort einfliessen.

Nachteile

Die Entwicklung der Software wird durch die Tests weitgehend unterbrochen, was zu etwas längeren Entwicklungszeiten führen kann. Auch hier finden Integrations- und Systemtests nach den regulären Sprints statt, somit wird der Zeitpunkt der Fehlererkennung nach hinten verschoben.

Empfehlung

Für Mittelgrosse ERP-Projekte empfohlen. Eine intensive Auseinandersetzung mit Integrations- und Systemtests ist für Projekte dieser Grösse existentiell. Mit der Anwesenheit der Entwickler/innen, Stakeholder und Fachexperten an den Test-Sprints sind die Kommunikationswege kurz. Entsprechend sind Fehlererkennung und Problemidentifikation einfacher und schneller. Fehleranalysen können bereits währen der Sprints stattfinden und entsprechend qualitativ in den Backlog einfliessen. Gewonnene Erkenntnisse können ohne weiteren Zeitverzug in die Entwicklung einfliessen.

4.3 Vorgehen, Methodik und Techniken

In einem agilen Projekt finden zu unterschiedlichen Zeitpunkten Testaktivitäten statt. In diesem Abschnitt soll aufgezeigt werden, welche Testaktivitäten zu welchem Zeitpunkt stattfinden und welche Methoden und Techniken dabei eingesetzt werden können.

4.3.1 Erstellung Backlog

Im Bereich Testing ist hier sicherzustellen, dass die erfassten Stories eine grundlegende Qualität aufweisen. Dazu kann das Akronym INVEST als Hilfestellung verwendet werden. Wichtig ist hier, dass die erfassten Stories testbar sind, respektive dass aus diesen Stories Testfälle abgeleitet werden können. Dieser Test kann der Product Owner durchführen, oder durch das 4-Augen Prinzip mit Personen aus den Entwicklungs-Teams.

Die Aufwändigste und sicherlich schwierigste Aufgabe ist die Definition der Features und Releases. Diese haben, wie bereits vorgängig beschrieben, einen grossen Einfluss auf die Teamzusammenstellung sowie auf die Testaktivitäten. Auf Basis des Release Plans werden im Verlaufe der Sprints die Integrations- und Systemtests (Feature-Tests) vorbereitet und terminiert.

Testaktivitäten:

- ✓ Quality Check User-Story
- ✓ Feature-Test-Plan erstellen

4.3.2 Vorbereitende Tätigkeiten

Um die Entwicklungs-Teams während der Sprints zu entlasten, können bereits einige Vorbereitungen getätigt werden. Im Bereich Dokumentation kann als erstes ein Storyboard erstellt werden, in welchem alle notwendigen Dokumente und Informationsflüsse skizziert werden. Auf Basis dieses Storyboards können anschliessend einige Vorlagen für beispielsweise die Dokumentationen von Testfällen erstellt werden. So kann das Entwicklungs-Team zu einem späteren Zeitpunkt auf diese Vorlagen zurückgreifen und benötig keine Zeit für Formatierungen und Dokumenten-Layout. Weiter kann sich das Projektteam bereits zu diesem Zeitpunkt Gedanken zu Testautomatisierungs- und Testdokumentations-Werkzeugen machen. Auch die Zusammenstellung der Teams sollte bereits vorgängig stattfinden. Auf der einen Seite ist es eine optimale Zusammenstellung der Entwicklungs-Teams mit den notwenigen Personas, auf der anderen Seite ist es das Feature-Test-Team oder das erweiterte Scrum-Team, welches mit den notwendigen Spezialisten ausgestattet werden muss. Auch die Testmaster-Persona ist vor den effektiven Sprints zu ernennen.

Testaktivitäten:

- ✓ Storyboard entwerfen
- ✓ Dokumentations-Templates erstellen
- ✓ Tool Testdokumentation evaluieren
- ✓ Tools Testautomatisierung ermitteln
- ✓ Teams definieren
- ✓ Testmaster Persona ernennen

4.3.3 Sprint Planning

Hier entstehen die ersten Testfälle. Durch das Scrum-Team ist sicherzustellen, dass für jede Anforderung mindestens ein Testfall definiert wird. Durch die Definition der Akzeptanzkriterien und den dazugehörigen Akzeptanztest auf Stufe Story entsteht die erste Grundlage für das Entwicklungs-Team. Zusätzlich könnten hier die Definition of Done für alle Stories festgelegt werden. Mit der Übergabe von einigen Praxisbeispielen (Specification by Example) können zudem reale Testszenarien von den Stakeholdern übergeben werden.

Testaktivitäten:

- ✓ Akzeptanzkriterien definieren
- ✓ Akzeptanztests auf Stufe Story entwerfen
- ✓ Definition of Done festlegen
- ✓ Spezification by Example

4.3.4 Sprint 0

Sofern ein Sprint 0 durchgeführt wird, können hier unterschiedliche Aufgaben im Bereich Testing durchgeführt werden. Optional können diese Aufgaben auch als Vorbereitende Tätigkeiten angesehen werden oder sogar in regulären Sprints umgesetzt werden, je nach Präferenz und Philosophie der Organisation. Im Bereich Testing ist hier sicher zentral, dass die Rollen und Verantwortungen im Bereich Testing besprochen und definiert werden. Insbesondere wenn ein Feature-Test-Team eingesetzt wird, und entsprechend in separaten Sprints oder sogar in separaten Teams getestet wird, müssen die Meetings und Informationsflüsse gut koordiniert sein. Der Informationsfluss zwischen den Teams wird für die Effizienz der Testaktivitäten entscheidend sein.

Des Weiteren kann im Sprint 0 auch die Technische Infrastruktur vorbereitet werden. So können die Teams beispielsweise Testserver oder allfällige Testwerkzeuge installieren.

Testaktivitäten:

- ✓ Testumgebung aufsetzten
- ✓ Test Tools installieren
- ✓ Test Tools Schulung
- ✓ Rollen und Verantwortung Test definieren

- ✓ Guideline Test Reporting erstellen
- ✓ Guideline Testdokumentation erstellen

4.3.5 Sprint 1-x

Auf Basis der erstellten Stories und Akzeptanzkriterien können die Teams nun mit der Entwicklung der Software beginnen. Die Testaktivitäten fokussieren sich hier stark auf die in diesem Sprint zu erledigenden Stories. Sofern Individualentwicklungen gemacht werden, könnte hier TDD eingesetzt werden. Des Weiteren können hier weitere Testfälle zu den Stories entworfen werden, sofern die Akzeptanzkriterien und die dazugehörigen Akzeptanztests noch nicht genug detailliert erstellt wurden. Sämtliche Testaktivitäten, welche zur Erfüllung der Story dienlich sind, können hier durch das Team selbständig durchgeführt werden.

Am Ende des Sprints sollte ein Austausch mit der Testmaster-Persona stattfinden. In diesem Austausch können die bereits durchgeführten Tests dem Feature-Test-Team übergeben werden, damit diese beispielsweise in automatisierten Regressionstest oder im Kontext von Feature-Tests weiterverwendet werden können.

Parallel zu den regulären Sprints kann das Feature-Test-Team oder der Testmaster (je nach Konstellation) bereits mit der Vorbereitung der Integrations- und Systemtests beginnen. So können auf Basis des Feature-Test-Plans und dem Austausch am Ende der jeweiligen Sprints bereits Testentwürfe erstellt und dokumentiert werden. Auch Hansmann (2020) empfiehlt hier, die Testfälle bereits vor den effektiven Testaktivitäten zu erstellen.

Testaktivitäten:

- ✓ Unit- und Komponententests entwerfen
- ✓ Unit-Test durchführen
- ✓ TDD (sofern sinnvoll)
- ✓ ATDD / BDD (teilweise, sofern sinnvoll)
- ✓ Testübergabe an Feature-Test-Team
- ✓ Akzeptanztests auf Stufe Feature entwerfen / dokumentieren

4.3.6 Feature-Test / Feature-Test-Sprints

Hier werden die Integrations- und Systemtest durchgeführt. Je nach Stand wird der bereits entwickelte Release für die Auslieferung freigegeben. Die bereits vorgängig definierten Feature-Tests können zu Beginn der effektiven Testsessions hier noch verfeinert werden. Als Ergänzung zu den Integrations- und Systemtest kann das Test-Team im Rahmen der erstellten Features zusätzliche Explorative Tests durchgeführt.

Durch ein angemessenes Reporting ist der Stand der Tests sowie die Qualität der Software einfach und schnell nachzuvollziehen. Gefundene Fehler werden so dokumentiert, dass das Entwicklungs-Team nachfolgend den Fehler einfach nachvollziehen beheben kann. Am Ende des Test-Sprints werden repräsentative Testfälle so dokumentiert, dass sie nach Auslieferung der Software der Betriebsorganisation übergeben werden können.

So können diese Testfälle zu einem späteren Zeitpunkt, zum Beispiel eines Softwarereleases auf Seite des ERP-Systemanbieters, wiederverwendet werden.

Testaktivitäten:

- ✓ Akzeptanztests auf Stufe Feature definieren / durchführen
- ✓ Explorative Tests durchführen
- ✓ Fehler Dokumentieren (Fehler Backlog)
- ✓ Repräsentative Testfälle zur Übergabe an Betriebsorganisation vorbereiten
- ✓ Fehlerbericht (Fortschritt und Qualität)

4.4 Testfallerstellung und Dokumentation

Bei der Testfallerstellung und Dokumentation muss zwischen den Entwicklungs-Teams und dem Feature-Test-Team unterschieden werden. Bei den Entwicklungs-Teams liegt der Fokus klar auf der erfolgreichen Umsetzung von Stories. Innerhalb der Sprints ist das Scrum-Team relativ frei in der Handhabung und Dokumentation der Tests. Hier sollten von Seite des Managements auch nicht viel Einfluss genommen werden (Bucka-Lassen, 2020). Mit einer durchdachten «Definition of Done» sowie gut formulierten Akzeptanzkriterien je Story sind die Vorgaben für die Scrum-Teams grundsätzlich klar. Wo sicherlich ein Augenmerk gesetzt werden muss, ist bei der Übergabe der erledigten Stories. Dies ist ein idealer Zeitpunkt um mit dem Feature-Test-Team in Kontakt zu treten. Dort können bereits durchgeführte Tests besprochen und semi-formal dokumentiert werden, sofern sich diese Tests auch für Integrations- und Systemtests eignen. Es ist auch zu besprechen, welche Komponententests bereits durch das Scrum-Team durchgeführt wurden, und welche nicht. Aus Sicht der frühen Fehlererkennung ist es sicherlich wünschenswert, dass die Scrum-Teams auch einige Komponententests durchführen, sofern in einem Sprint mehrere Units zu einer Komponente erstellt wurden. Ansonsten können Testaktivitäten rund um Komponententests auch durch Feature-Test-Team durchgeführt werden. Hier besteht jedoch die Gefahr, dass die Fehler eher später identifiziert werden.

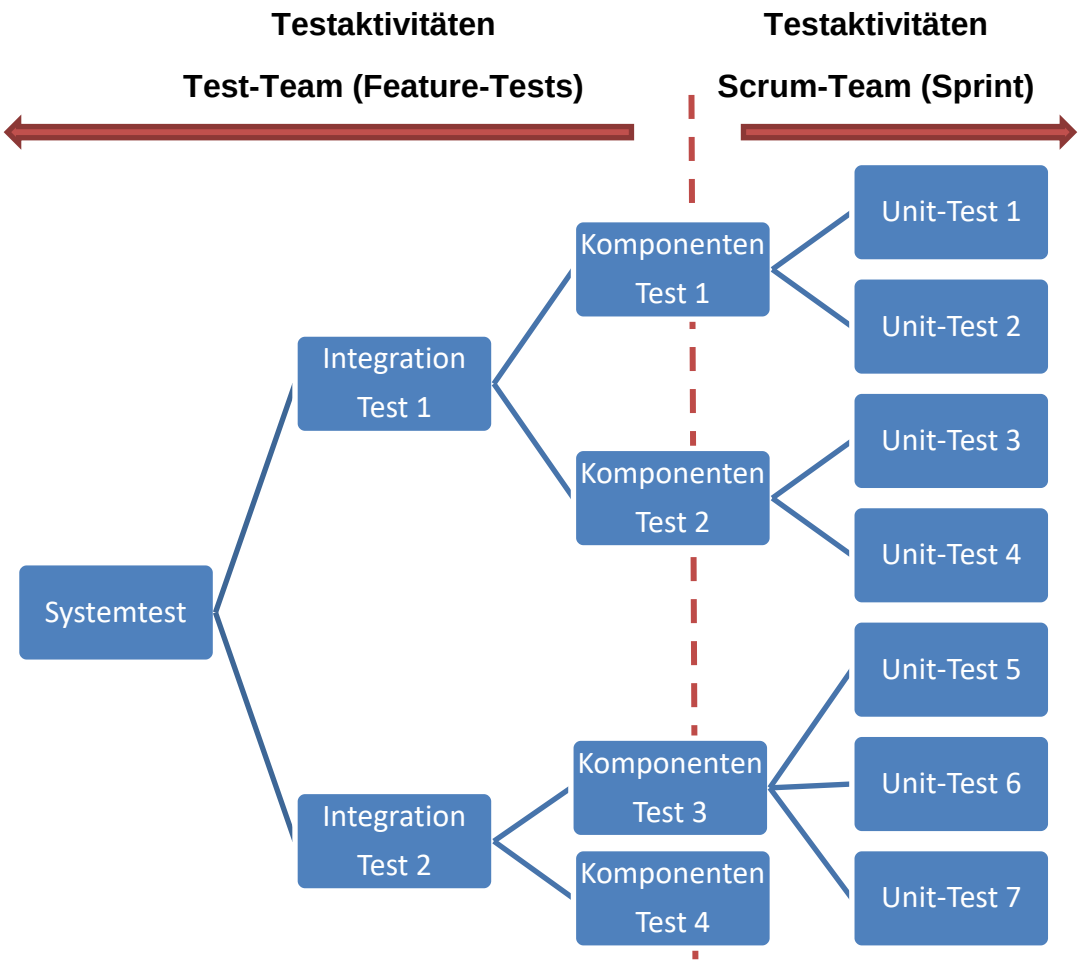

Abbildung 36:Verantwortung Testaktivitäten Scrum-Team und Test-Team

Quelle: Eigene Darstellung

Betreffend Form und Art der Dokumentation sollte sich die Organisation im Sprint 0 oder als vorbereitende Tätigkeiten auf geeignete Templates einigen. Welche Dokumente gefordert werden, ist vorab in einem Storyboard zu definieren (siehe Storyboard 2.3.1) «Gold Plating» zu betreiben ist sicherlich nicht sinnvoll und im agilen Ansatz auch nicht gewünscht. Ein mögliches Beispiel für eine eher ausführliche Testfallbeschreibung von Feature- und Integrationstests ist in der nachfolgenden Tabelle abgebildet:

Testfall	Name / Eindeutige ID
Testobjekt	Beschreibung der zu testenden Anwendung
Testkonfiguration/ Testdaten	Welche Systemumgebung und welche Umgebungsvoraussetzungen sind Voraussetzungen für den Test? Verweis auf die im Test notwendigen Testdaten und Rahmenbedingungen.
Testbeschreibung	Genaue Beschreibung des Testfalls und der Schritte, die bei der Ausführung zu beachten sind.

Bezug	Zu welcher Story / Feature gehört dieser Testfall? Welchen Bezug hat dieser Testfall zu anderen Testfällen?
Priorität	Zwingend erforderlich oder eher untergeordnet
Details	Weitere Details zum besseren Verständnis.
Soll Ergebnis	Welches Ergebnis wird erwartet?
Ist-Ergebnis	Welches Ergebnis wurde erzielt?
Bestanden	War der Test erfolgreich?
Fehlerursache	Falls ein Fehler vorliegt: Wo ist der Fehler aufgetreten? Wie hat sich das System verhalten?
Kommentar	Weitere Kommentare
Tester/in	Wer hat den Test durchgeführt? (für Rückfragen)
Datum / Uhrzeit	Wann wurde der Test durchgeführt

Tabelle 4: Beschreibung eines Testfalls

Quelle: Witte (2019, S. 162)

Die Wahl der Dokumentationssoftware ist aus Sicht unterschiedlicher Experten zweitrangig (Frei, 2020) / (Hansmann, 2020). Ein Werkzeug mit Mehrbenutzerfähigkeit wird jedoch empfohlen, da ansonsten ein Wildwuchs an Dokumenten droht (Witte, 2019, S. 160).

An diesem Punkt ist noch wichtig zu erwähnen, dass ist die Zuordnung der Testfälle zu Features oder Prozessen (Witte, 2019, S. 159) gut überlegt werden soll. Eine sinnvolle Kategorisierung und Identifikation der Testfälle ist im Rahmen einer ERP-Einführung wichtig, sowohl für die Fehlerverfolgung als auch für die Übergabe der Testfälle an den Betrieb.

4.5 Fehlertracking und Reporting

Wie bereits vorgängig erwähnt, ist innerhalb der Entwicklungs-Teams kein grosses Augenmerk auf das Fehlertracking und Reporting zu legen. Entweder ist eine Story fertig und hat entsprechend die Akzeptanzkriterien sowie die Definition of Done erfüllt, oder eben nicht. Wenn Fehler innerhalb der Sprints behoben werden, gibt es grundsätzlich auch keine, die dokumentiert werden müssen. Testen und entsprechende Fehler gehört zur Entwicklung dazu. Kann ein Fehler nicht behoben werden, muss die Story im nächsten Sprint nochmals in den Sprint-Backlog oder es wird notfalls eine neue Story erfasst, welche nur die Behebung des Fehlers beinhaltet. Dies kann je nach Ausmass des Fehlers variieren. Feature-Test-Teams haben hier eine andere Rolle. Gefundene Fehler

können nicht sofort behoben werden, entsprechend müssen hier die Fehler strukturiert erfasst und anschliessend in den Backlog einfliessen.

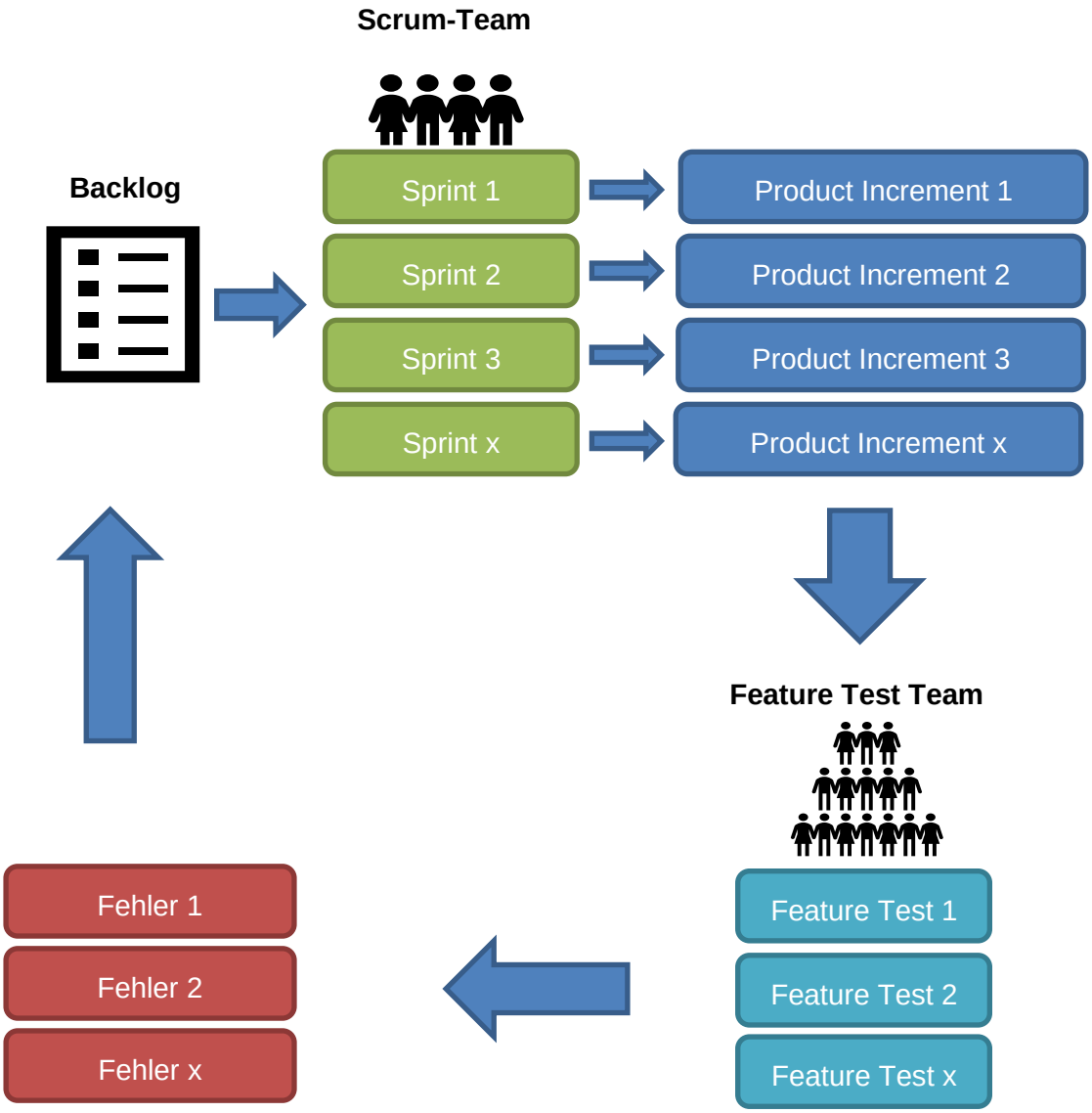

Abbildung 37: Fehlertracking mit Feature-Test-Team

Quelle: Eigene Darstellung

Mit diesem Vorgehen ist sichergestellt, dass die Behebung des Fehlers mindestens durch das Entwicklungs-Team, im Normalfall auch durch das Test-Team (Übergabe der Story z.H. Feature-Test-Team) erneut getestet wird. Hier wird empfohlen, einen Standard-Aufbau für die Fehlermeldung zu definieren. Auch hier ist «Gold Plating» nicht sinnvoll. Eine einfache und präzise Beschreibung des Fehlers ist ausreichend. Diese Vorlage könnte ebenfalls im Sprint 0 erstellt und gemeinsam festgelegt werden. Witte (2019) beschreibt, dass eine Fehlermeldung wie folgt aufgebaut werden kann (S. 103):

Problembeschreibung	Beschreibung der aufgetretenen Fehler im Verlaufe der Tests
Identifikation	Test ID, Testperson, Feature, Testumgebung (Rahmendaten)
Klassifikation	Fehlerklasse und Priorisierung

Tabelle 5: Aufbau einer Fehlermeldung

Quelle: Witte (2019, S. 103)

Ein spezielles Augenmerk sollte auf die Fehlerklassifikation gelegt werden (Witte, 2019, S. 103). Diese ist einerseits relevant, ob der Release freigegeben werden kann und mit wie hoch dieser Fehler im Backlog priorisiert werden soll. Die Informationen aus der Fehlermeldung sollten 1:1 in der Story enthalten sein. So kann der Abklärungsaufwand währen des Sprint Planning minimiert werden und bei erneuten Tests kann der Testfall als «Bestanden» markiert werden. So schliesst sich der Kreis wieder und einfaches und effizientes Reporting wird ermöglicht.

Im Bereich Reporting gäbe es unzählige Kennzahlen, welche eingesetzt werden können, von der Testprozessreife, über Testproduktivität bis hin zu Prozesseffektivität und Aufwandsberechnungen (Witte, 2019, S. 143-150). Der Nutzen dieser Kennzahlen hält sich jedoch in Grenzen, wenn diese nicht dazu genutzt werden, um die Testaktivitäten aktiv zu steuern. Solche Kennzahlen machen bei professionellen Softwareentwicklungs-Unternehmen vermutlich eher Sinn, um die Testaktivitäten langfristig zu optimieren. Eine ERP-Einführung ist in ihrer Ausprägung jedoch eher einmalig. Zudem ist Kennzahlen «auf Vorrat» zu erstellen nicht zielführend und effektiv. Sie machen entsprechend nur bedingt Sinn. Aus diesem Grund wird hier empfohlen, möglichst einfache und aussagekräftige Metriken zu verwenden. Bei Bedarf können diese immer noch erweitert werden. Aus den nachfolgenden Messgrössen wurden anschliessend unterschiedliche Beispieldiagramme erstellt:

- Anzahl erstellte Testfälle (je Feature)
- Anzahl durchgeführte Testfälle (je Feature)
- Anzahl bestandene Testfälle (je Feature)
- Fehlerkategorisierung (je Testfall)

Im Diagramm (Abbildung 38) kann man den Testfortschritt je Feature beurteilen. Es wird die Anzahl erstellter Testfälle mit der Anzahl durchgeführten Testfälle verglichen. So ist schnell ersichtlich, wie weit die Tests vorangeschritten sind.

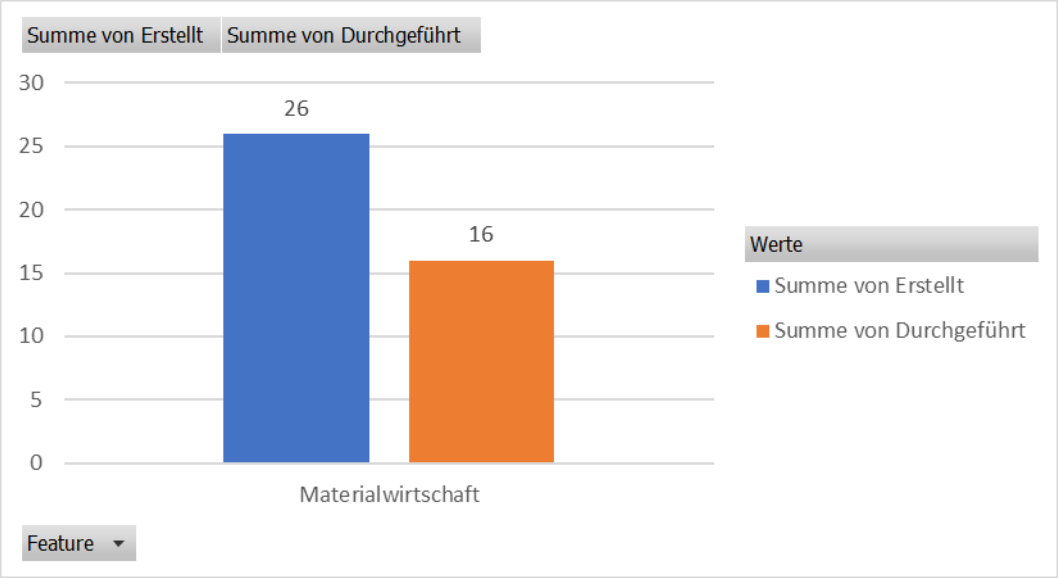

Abbildung 38: Übersicht Testfortschritt je Feature

Quelle: Eigene Darstellung

Mit der Gegenüberstellung der Anzahl durchgeführter Tests und der Anzahl bestandener Tests (Abbildung 39) kann beurteilt werden, wie viele Fehler bei den Testaktivitäten entdeckt wurden respektive wie viele Testfälle nicht bestanden wurden:

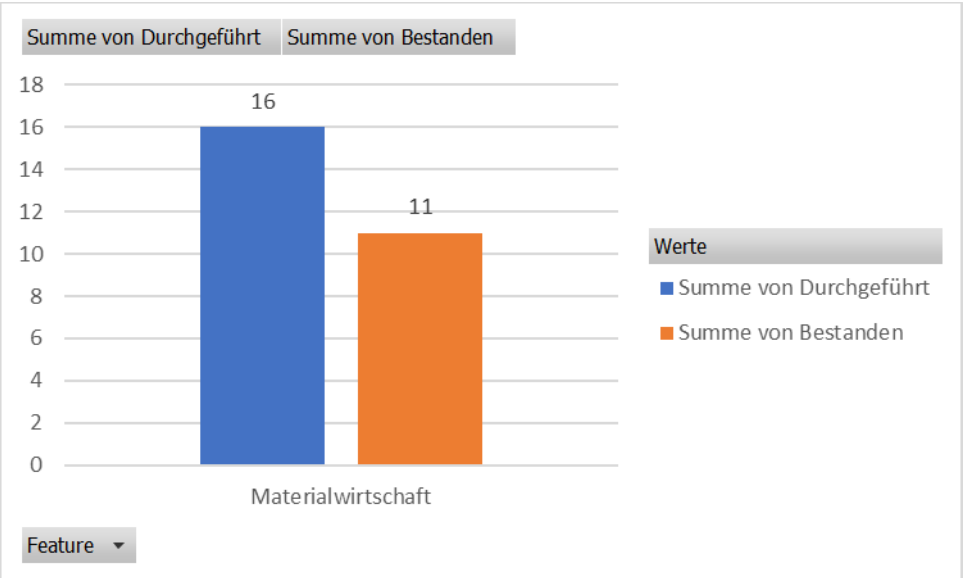

Abbildung 39: Übersicht Bestandene Testfälle je Feature

Quelle: Eigene Darstellung

Um die Auswirkung der Fehler besser zu beurteilen, können die Fehler mit einer Fehlerklasse angereichert werden. So ist relativ einfach zu ermitteln, wie gross sich die gefundenen Fehler auf die bereits erstellten Softwarekomponenten auswirken würden. Dies ist ebenfalls ein weiteres Hilfsmittel für den Entscheid, ob ein Release ausgeliefert wird oder nicht.

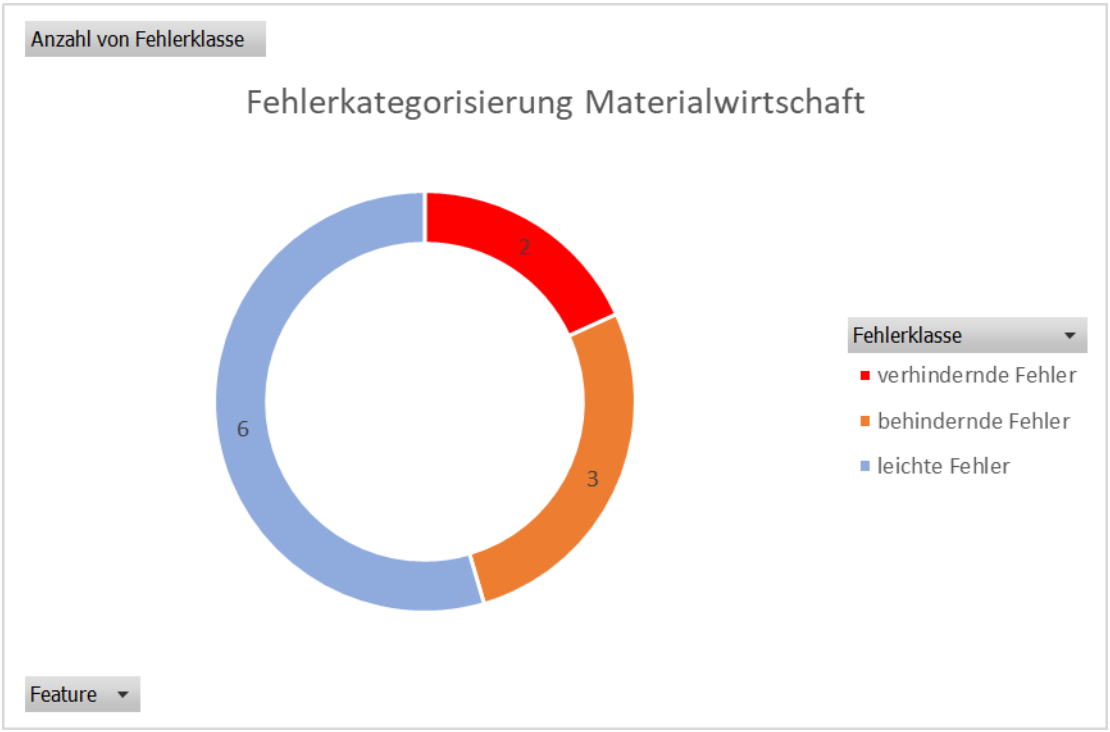

Abbildung 40: Kategorisierung der Fehler je Feature

Quelle: Eigene Darstellung

Mit Hilfe dieser drei Grafiken können bereits gute Aussagen zur Qualität der Software und zum Stand der Tests gemacht werden. Natürlich können diese Grafiken noch beliebig mit Zeitreihen ergänzt werden, um auch den Fortschritt nach mehrerer Testrunden zu beurteilen. Weitere Kennzahlen werden in einem ersten Schritt nicht empfohlen. Die Test-Teams sollten ihre wertvolle Zeit mit Testen verbringen und nicht mit dem nachführen und auswerten von Kennzahlen.

4.6 Testautomatisierung

Experten in der agilen Welt sprechen sich sehr für Testautomatisierung aus und sehen darin einen klaren Nutzen im Bereich der Qualität sowie der Entwicklungszeiten. Da in ERP-Projekten im Vergleich zu klassischer Softwareentwicklung wenig Code programmiert wird, muss der Nutzen von solchen Werkzeugen hinterfragt werden. Unter der Voraussetzung, dass das eingesetzte ERP-System zu Beginn der Sprints bekannt ist, können vorgängig einige Abklärungen getroffen werden:

- Welche Personen werden im Projekt hauptsächlich testen?
- Existieren auf dem Markt Testwerkzeuge, welche bei dem eingesetzten ERP-System automatisiert Stammdaten erzeugt werden können?
- Existieren auf dem Markt Testwerkzeuge, welche bei dem eingesetzten ERP-System automatisiert Prozesse durchgeführt werden können?
- Sind Empfehlungen von Seite ERP-Anbieter bezüglich Testwerkzeugen vorhanden? Eignen sich diese in der agilen Umsetzung?
- Gibt es bei dem eingesetzten ERP-System bereits integrierte Werkzeuge, die für die Erfassung von Stammdaten und Testfälle verwendet werden können?
- In welchen Features ist mutmasslich der grösste Entwicklungsaufwand, und welche Herausforderungen bezüglich der Tests könnten dort liegen.

Nach der Auseinandersetzung mit den oben erwähnten Fragestellungen kristallisiert sich möglichweise eines oder mehrere präferierte Testwerkzeuge heraus. Diese könnten in Form einer Produkte-Demo oder Testlizenz genauer angeschaut werden. Ebenso machen sich die Personen vorab Gedanken, wo die Herausforderungen im Testing liegen könnten. Aufgrund der gewonnenen Erkenntnisse können im Sprint 0 bereits erste Werkzeuge zu Pilotzwecken installiert werden. In den nachfolgenden Sprints können diese Werkzeuge ausführlich getestet werden. Es ist auch denkbar, dass in unterschiedlichen Scrum-Teams unterschiedliche Werkzeuge eingesetzt werden. Jedoch ist hier sicherlich ein enger Austausch zwischen den Teams sinnvoll. So können gewonnene Erkenntnisse ausgetauscht und die effektive Eignung der verschiedenen Werkzeuge ermittelt werden. Sofern die Testpersonen einen Nutzen in diesen Werkzeugen sehen, kann der Pilotbetrieb ausgebaut werden, auch mit mehreren Werkzeugen, sofern diese unterschiedlichen Zwecken dienen. Erweisen sich die Werkzeuge als wenig effizienzsteigernd, so kann der Pilot abgebrochen werden. Bei dem Entscheid für oder gegen Testautomatisierung ist jedoch zu beachten, dass der Testaufwand im Verlaufe der Sprints laufend zunimmt, da bestehende Entwicklungen in Form von Regressionstests laufend mitgetestet werden sollten.

4.7 Testen aus integrativer Sicht

In Kapitel 3.6 Systemintegration/Schnittstellen wurde erwähnt, dass Schnittstellen zu Umsystemen ein nicht zu unterschätzendes Element in der Einführung von ERP Systemen darstellt. Ebenso wurde erwähnt, dass die Bildung von Features sowie deren Zuteilung an Scrum-Teams äusserst wichtig ist. So wird empfohlen, dass die Features möglichst so gestaltet sind, dass eine Überschneidung der Tests zwischen den Teams möglichst vermieden wird. Dies ist vor allem bei der Entwicklung von Schnittstellen äusserst empfehlenswert. Zum einen wird der Spezialist für die anzubindende Applikation so nur in einem Team eingesetzt und es entstehen entsprechend keine Ressourcenkonflikte zwischen den Teams. Zum anderen sind Integrationstest weitgehend durch ein Team zu bewältigen, und es muss entsprechend keine vertiefte Koordination zwischen den Teams stattfinden.

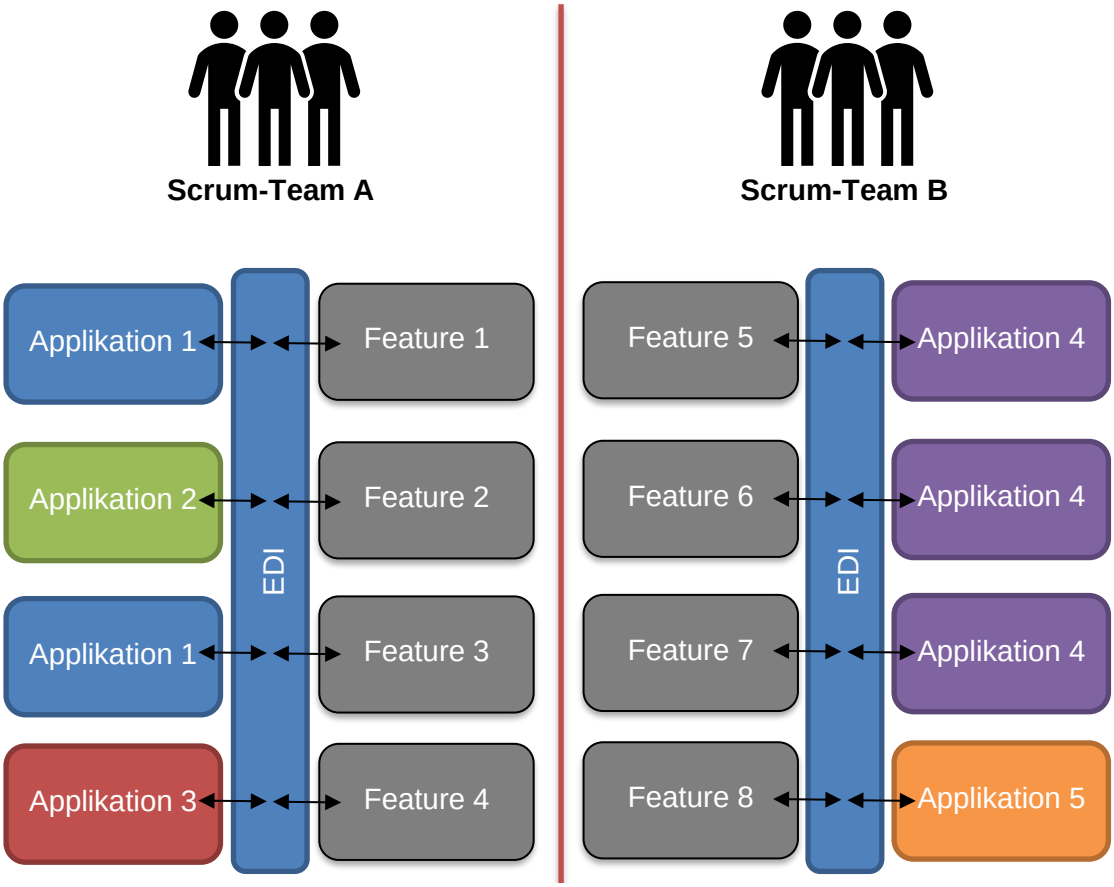

Abbildung 41: Zuteilung der Features nach Schnittstellen

Quelle: Eigene Darstellung

Was bei dieser Konstellation sicherlich zu beachten ist, dass Schlüsselressourcen wie beispielsweise Schnittstellenentwickler/innen in beiden Teams vorhanden sind. Ansonsten müssen die Sprints so geplant werden, dass die Ressourcen nicht gleichzeitig benötigt werden.

Obwohl bei der agilen Entwicklung grundsätzlich diejenigen Stories umgesetzt werden, welche den höchsten Business Value ausweisen, ist dies bei der Entwicklung von

Schnittstellen vermutlich mit etwas Zurückhaltung anzuwenden. Diese Tests sind erfahrungsgemäss eher aufwändig, auch wenn ein Teil der Tests automatisiert werden könnte. So ist in der Releaseplanung in Erwägung zu ziehen, die Schnittstelle zu einer Applikation möglichst in ihrer kompletten Endausprägung zu entwickeln. Natürlich ist dies nicht immer möglich, da in der agilen Entwicklung die Architektur noch nicht zu beginn komplett feststeht. Jedoch muss beachtet werden, dass jede Anpassung der Schnittstelle aufwändige Tests zur Folge haben. Gemäss dem Sprichwort «never change a running system» könnten hier einige agile Grundsätze etwas weiter ausgelegt werden, als von den Erfindern ursprünglich angedacht. Frei (2020) erwähnt zu diesem Punkt, dass der Business Value jeweils in Kombination mit dem Risiko betrachtet werden soll. So werden komplizierte und kritische Schnittstellen automatisch höher priorisiert.

Bezüglich der Erstellung und Dokumentation von Testfällen ist hier ein spezielles Augenmerk zu legen. Wo in ERP-Systemen Standardfunktionalitäten bereits auf Seite der ERP-Anbieter mehr oder weniger ausführlich getestet wurden, liegt bei der Entwicklung und dem Betrieb von Schnittstellen die Verantwortung weitestgehend beim Kunden respektive beim Scrum-Team. Schnittstellen müssen entsprechend ausführlich dokumentiert sein. Auch bei der Dokumentation der Testfälle kann hier ruhig etwas mehr investiert werden. Denn diese Testfälle werden nachfolgend zu unterschiedlichen Zeitpunkten wieder benötigt:

- Währen der Sprints bei Unit-, Komponenten- und Integrationstests
- Bei Systemtests vor der Produktfreigabe
- Nach Einführung, bei Releasewechsel von Seite ERP-Systemanbieter
- Bei Releasewechsel oder Anpassungen von angebundener Applikation
- Bei internen Systemanpassungen und Individualentwicklungen

Zusammenfassend kann hier folgendes empfohlen werden:

- Entwicklung einer Schnittstelle, wenn immer möglich durch dasselbe Team durchführen
- Applikationsspezialisten sollten während der Dauer der Schnittstellenentwicklung im Bestandteil des Scrum-Teams sein
- Die Schnittstelle möglichst in ihrer vollen Ausprägung und fokussiert entwickeln
- Nach jeder Anpassung ausführliche Tests in allen betroffenen Systemen durchführen
- Testfälle ausführlich dokumentieren und am Ende des Projektes der Betriebsorganisation übergeben

5 Fazit

Dieser Abschnitt dient dazu, die Ergebnisse dieser Arbeit zu beurteilen und das Vorgehen des Autors kritisch zu hinterfragen. Es soll zusätzlich interessierten Personen aufzeigen, wie die Ergebnisse zu werten sind und in welcher Form diese in die Praxis verwendet werden können.

5.1 Forschungsfragen

Zu Beginn dieser Masterthesis wurden Forschungsfragen formuliert, welche den Inhalt dieser Masterarbeit in wesentlichem Masse beeinflussten. In diesem Kapitel wird kurz darauf eingegangen, wie diese Forschungsfragen beantwortet wurden, und wo möglicherweise noch einige Lücken bestehen.

Frage 1

Welchen Einfluss hat Agiles Vorgehen in einem ERP-Projekt auf das Testing?

Der Beantwortung dieser Frage wurde viel Bedeutung geschenkt. Es wurden die Unterschiede zwischen agilem Testing und dem ursprünglichen Vorgehen in Wasserfall-Projekten aufgezeigt. Schritt für Schritt wurde erläutert, welche Aspekte in Scrum Projekten im Bereich Testing relevant sind.

Frage 2

Welche organisatorischen Voraussetzungen müssen in einem Unternehmen für ein erfolgreiches Testing im einem agilen ERP-Projekt erfüllt werden?

Auch diese Frage wurde detailliert beantwortet. Die Zusammenstellung der unterschiedlichen Teams ist hier der entscheidende Faktor. Zu diesem Zweck wurden Personas mit unterschiedlichen Fähigkeiten definiert, welche in den Teams eingesetzt werden sollten. Zusätzlich wird die Definition von Features zusätzlich einen wesentlichen Einfluss sowohl auf die Entwicklung als auch auf das Testing haben. Auch die Verfügbarkeit von Ressourcen wird, wie in allen Projekten, über Erfolg und Misserfolg entscheiden.

Frage 3

Wie wird das Testmanagement konzipiert und welche Methoden und Techniken eignen sich in einem agilen ERP-Projekt?

Die Beantwortung dieser Frage ist in einem ERP-Projekt eher schwierig. Die in der Literatur erwähnten Techniken beziehen sich eher auf die Softwareentwicklung als auf die Einführung von Software. Auch Testautomatisierung wird in einem ERP-Projekt eher schwierig zu betreiben sein. Somit konnten hier keine präzisen Empfehlungen gemacht werden. Mit der Auflistung unterschiedlicher Testaktivitäten und verschiedenen Elementen von Scrum konnte jedoch eine gute Übersicht über die Testaktivitäten in den verschiedenen Elementen des Scrum Frameworks erstellt werden.

Frage 4

Wie kann die Qualität der Software aus integrativer Sicht mit produktiv laufenden Umsystemen sichergestellt werden?

Zur Beantwortung dieser Frage wurden auch unterschiedliche Experten zu Rate gezogen. Eine Unterscheidung von generellen Tests und solchen welche Schnittstellen betreffen war aus Sicht der Experten nicht notwendig. Obwohl die Entwicklung der Schnittstellen aufwändig ist, unterscheiden sich die Tests nicht wesentlich von den restlichen. Die Aufteilung der Schnittstellen-User-Stories auf verschiedene Scrum-Teams wird hier wohl den weit grösseren Einfluss haben.

Frage 5

Wie werden Testfälle und Ergebnisse dokumentiert damit sie nach Projektabschluss in den operativen Betrieb überführt werden können?

Hier wurde aufgezeigt, zu welchem Zeitpunkt Testdokumentationen erstellt werden sollten und wie diese auch dem Betrieb übergeben werden. Mit dem aufzeigen von einigen Beispielen, wie beispielsweise das Storyboard, Testfallbeschreibungen oder Fehlerberichten kann hier ein guter Praxisbezug hergestellt werden.

5.2 Zielsetzungen

In diesem Abschnitt wird kurz auf die von den Forschungsfragen abgeleiteten Zielsetzungen eingegangen und beurteilt, wie diese erfüllt wurden:

Ziel 1

Es wird ein mögliches Vorgehen für die Erstellung einer Testkonzeption in einem agilen ERP-Projekt beschrieben.

Dieses Ziel wurde aus Sicht des Autors erreicht. Es konnten klare Empfehlungen verfasst werden, zu welchem Zeitpunkt unterschiedliche Testaktivitäten stattfinden und welche Aufgaben von welchen Personen und Teams wahrgenommen werden sollten. Es ist ersichtlich, wie Testing in agilen ERP-Projekten aufgebaut werden kann, und welche Faktoren erfolgsrelevant sind.

Ziel 2

Es wird aufgezeigt, wie Testing in einer agilen ERP-Projektorganisation eingebettet werden kann.

Im Rahmen dieser Zielsetzung wurden unterschiedliche Optionen aufgezeigt. Es stellte sich heraus, dass in agilen ERP-Projekten separate Testteams notwendig sind, obwohl dies im Scrum-Framework nicht vorgesehen wäre. Diese Erkenntnis ist sicherlich sehr wertvoll für die Zusammenstellung der Teams in einem ERP-Projekt.

Ziel 3

Es sind geeignete Massnahmen bekannt, wie die Qualität der Integration von produktiv laufenden Umsystemen sichergestellt wird.

Für diese Zielsetzung wurden mehrere Möglichkeiten zur Reduktion von Komplexität aufgezeigt. Diese Massnahmen sind jedoch eher organisatorischer als technischer Natur. Auf technische Massnahmen, wie beispielsweise Testautomatisierungen oder spezialisierte Werkzeuge wurde nur am Rande eingegangen. Die technischen Möglichkeiten müssten hier wohl durch einen IT-Spezialisten erarbeitet und beurteilt werden und könnten Teil einer weiterführenden Untersuchung sein.

5.3 Reflexion des Autors

Mit der Erstellung dieser Masterarbeit habe ich sehr viel über Agilität, speziell über agiles Testen gelernt. Bereits diese Tatsache rechtfertigt den sehr hohen Aufwand für die Erstellung einer solchen Arbeit. Obwohl unterschiedliche Experten aus der Praxis befragt wurden, fehlt mir selbst die Praxiserfahrung noch weitgehend. Aufgrund dieser Tatsache wird die Verteidigung meiner Arbeit, gegenüber einem Experten mit viel Praxiserfahrung, zu einer Herausforderung. Die Ergebnisse aus dieser Arbeit sind entsprechend auch mit Vorsicht zu geniessen. Natürlich können aus der Literatur und Interviews mit Experten gute Schlüsse gezogen werden. Jedoch müssen sich die in dieser Arbeit gemachten Empfehlungen erst noch in der Praxis beweisen. Agile Vorgehensweisen sind zum Glück prädestiniert für solche Feldversuche. Lernen ist ein stetiger Prozess, welcher mit der Verfassung einer Masterthesis bei weitem nicht abgeschlossen ist. Es bleibt somit spannend.

Der Wissenschaftliche Aspekt meiner Arbeit ist aufgrund der Experteninterviews und der verwendeten Literatur sicherlich vorhanden, jedoch ist in diesem spezifischen Fachgebiet noch eher wenig qualifizierte Literatur vorhanden. Erschwerend kommt hinzu, dass sowohl die Autoren von Büchern als auch die Experten unterschiedliche Auffassungen von agilem Testing haben und gleichzeitig das Scrum-Framework sehr viel Interpretationsspielraum offenlässt. Es hat sich zudem aufgrund unterschiedlicher Gespräche herausgestellt, dass die Literatur im Bereich Scrum nicht immer optimal ist. Dessen Beurteilung ohne Praxiserfahrung erwies sich jedoch als schwierig.

Der selbst gesetzte Zeitplan für die Erstellung dieser Arbeit ist mit viel Fleiss und Disziplin aufgegangen. Erstmalig in meiner Zeit als Student konnte ich eine solche Arbeit erstellen, ohne dass Nachtschichten eingelegt werden mussten. Und dies nicht, weil weniger Aufwand investiert wurde, sondern weil ich meine restlichen Freizeitaktivitäten auf ein Minimum reduzierte und auch über mehrere Tage am Stück konsequent an dieser Thesis arbeitete. Durch die Vermeidung von Störungen, beispielsweise durch eine geschickte Wahl des Arbeitsplatzes, konnte ich eine zusätzliche Effizienzsteigerung erzielen.

Unabhängig von der Bewertung dieser Arbeit kann ich mit Stolz auf diese Zeit zurückschauen. Nicht jeder Mensch hat in seinem Leben den Aufwand für die Erstellung einer Masterarbeit auf sich genommen. Mit dem Wissen, dass in dieser Arbeit nicht in allen Aspekten Perfektion erreicht wurde, kann die eigene Einschätzung doch als Schlusswort vermerkt werden:

«on time, on budget, on scope, on quality»

6 Literaturverzeichnis

Acreo Consulting. (07. Februar 2020). Von https://www.acreo.ch abgerufen

Aragost. (07. Februar 2020). Von https://aragost.com/ abgerufen

Baumgartner, M., Klonk, M., Pilcher, M., Seidl, R., & Tanczos, S. (2018). *Agile Testing.* München: Carl Hanser Verlag.

Becker, M., & Ortmann, J. (21. 12 2019). *Marco Becker Management Consultants.* Von http://marco-becker.eu/sites/default/files/documents/ERP-Studie.pdf abgerufen

Bucka-Lassen, K. (13. Januar 2020). Interview mit Agile Coach. (N. Costantino, Interviewer)

Frei, D. (09. Januar 2020). Interview mit unabhängigem ERP- Berater. (N. Costantino, Interviewer)

Gloger, B. (2016). *Scrum.* München: Carl Hanser Verlag.

Gregory, J., & Crispin, L. (2015). *Agile Testing Collection.* New Jersey: Pearson Education, Inc.

Hansmann, R. (13. Januar 2020). Interview mit ERP-Projektleiter. (N. Costantino, Interviewer)

Heuer, F. (2014). *Agile Testing.* Unites States: CreateSpace Independent Publishing Platform

Hruschka, P. (2019). *Business Analysis und Requirements Engineering.* München: Carl Hanser Verlag.

Meier, H. (15. Januar 2020). Interview mit ERP-Systemanbieter. (N. Costantino, Interviewer)

Ritterkamp, C., Schmitz-Ohrndorf, M., Arndt, N., Harstick, M., & Knapp, D. (2013). *Agiles Requirements Engineering und Testen.* Frankfurt am Main*:* Software & Support Media GmbH.

Robson, S. (2016). *Agile SAP (Kindle Version).* Cambridgeshire: IT Governance Limited.

Rupp, C. (2014). *Requirements-Engineering und -Management.* München: Carl Hanser Verlag.

Schmelzer, H., & Sesselmann, W. (2013). *Geschäftsprozessmanagement in der Praxis.* München: Carl Hanser Verlag.

Scrum Academy. (08. Februar 2020). Von https://www.scrum-academy.de/product-owner/wissen/sprint-zero/ abgerufen

SwissQ Consulting AG. (11. Januar 2020). Von https://swissq.it/de/agile/agile-testing/swissq-agile-testing-framework/ abgerufen

Wikipedia. (26. Dezember 2019). Von Wikipedia: https://de.wikipedia.org/wiki/Martin_Luther_King abgerufen

Witte, F. (2019). *Testmanagement und Softwaretest.* Wiesbaden: Springen Fachmedien Wiesbaden GmbH.

Anhang A

Interview mit unabhängigem ERP- Berater

Thema: Agiles Testen in ERP-Projekten

Ziel des Interviews: Durch die Erfahrung von Experten soll das Testen in agilen ERP-Projekten genauer untersucht werden. Welche Vorgehensweisen haben sich aus Sicht ERP-Berater bewährt und was muss speziell beachtet werden.

Datum: 09.01.2020

Ort: St. Gallen

Art des Interviews: schriftlich

Interviewpartner: Daniel Frei

Funktion: Senior Project Manager

Geburtsdatum: 08.11.1974

Unternehmen: acreo consulting ag, 9000 St. Gallen

Generelles: Die nachfolgenden Fragen sind freiwillig und müssen nicht beantwortet werden. Auf Ihren Wunsch werden Informationen, welche auf Sie oder Ihr Unternehmen schliessen lassen, anonymisiert. Telefonisch- oder persönlich durchgeführte Interviews werden aufgezeichnet, damit sie paraphrasiert und in der Arbeit verwendet werden können. Aufzeichnungen werden nicht aufbewahrt.

Anonymität: Personen- und Unternehmensinformationen dürfen in der Arbeit vermerkt werden

1. Allgemeine Fragen zu agilen ERP-Projekten

1.1 Sehen Sie in den letzten Jahren eine Tendenz, dass ERP-Projekte vermehrt agil umgesetzt werden? Wenn Ja, welche agilen Methoden (z.B. Scrum, Kanban… usw.) werden dabei hauptsächlich angewendet?

Ja, die Tendenz ist erkennbar. Hauptsächlich Scrum und/oder einzelne Elemente wie Backlog / Inkrement / Timeboxing.

1.2 Welche **Vorteile** sehen Sie bei einer agilen ERP-Einführung im Vergleich zu herkömmlichen Vorgehensweisen (z.B. Wasserfall)?

Dass die mögliche Methodenvielfalt grösser wird.

1.3 Welche **Nachteile** sehen Sie bei einer agilen ERP-Einführung im Vergleich zu herkömmlichen Vorgehensweisen (z.B. Wasserfall)?

Das «Agilität» nur dort angewendet (oder auch nur erwähnt) wird, wo es gerade besser passt.

1.4 Wenn wir nun annehmen, dass Scrum die präferierte agile Einführungsmethodik ist, wie würden Sie ein ERP-Scrum-Team aus Ihrer Sicht zusammenstellen? Welche Fähigkeiten sollten die Teammitglieder aus Ihrer Sicht besitzen?

So umfassend und evtl. auch divergent wie möglich. Dies in Bezug auf fachliche Expertise, auf bspw. Alter / Geschlecht, auf Stakeholder Interessen, …

1.5 Aufgrund theoretischer Aspekte sowie der eigenen Erfahrung aus ERP-Projekten habe ich die Schlussfolgerung gezogen, dass alle Tests (inkl. Integrations- und Systemtests) möglichst durch das Scrum-Team selbständig durchgeführt werden sollten. Entsprechend müssen die dazu notwendigen Fachpersonen, wenn immer möglich Bestandteil des Scrum-Teams sein. Was halten Sie von dieser Aussage?

Ich sehe das Scrum-Team ebenfalls in der Verantwortung zur Abnahme der operativen Tests (Features, Abnahmen, …). Mir scheint jedoch wichtig zu sein, dass die Koordination (auch strategisch), die Konzeption (auch strategisch) und die Umsetzung mit getrennten Verantwortlichkeiten organisiert wird.

2. Fragen zum Testing in (agilen) ERP Projekten

2.1 Wo / bei welchen Aktivitäten liegt aus Ihrer Sicht der grösste Testaufwand in ERP-Einführungsprojekten?

Test Cases und Abläufe definieren

2.2 Wie hoch (Prozentsatz von total aufgewendeter Zeit) schätzen Sie den Aufwand für das Testing bei einer ERP-Einführung?

Eine Antwort in % scheint mir schwierig, da dies sehr vom prozentualen Anteil der Entwicklungsaufwände abgängig ist. Eine standardnah eingesetzte ERP Lösung mit wenig Features wird vielleicht 5 – 10% für das Testing einsetzen. Auf der anderen

Seite kann ich mir auch Aufwände über 50% vorstellen. Dies war bspw. bei einem Projekt im Flugsicherungsumfeld der Fall, dabei haben wir ca. 50% aller aufgewendeter Zeit für das sich wiederholende Testen eingesetzt.

2.3 Wie, in welcher Form und zu welchem Zweck empfehlen sie Ihren Kunden die Dokumentation von Testfällen? Können Sie den Einsatz von Testdokumentations-Werkzeugen wie Beispielsweise Jira empfehlen?

Die Form erachte ich als zweitrangig, wichtig ist die Test Cases und die Abläufe nachvollziehbar, transparent und eindeutig zu dokumentieren. Ich schätze vor allem den Einsatz von den Testbeteiligten bereits bekannten Werkzeugen. So kann man sich als Team auf das wesentliche konzentrieren und beschäftigt sich nicht mit administrativen Tools. Bei grösseren Projekten habe ich gute Erfahrungen mit Jira gemacht, dies setzt jedoch eine Vorkenntnis der Beteiligten voraus.

2.4 Die Rolle als Testmanager gibt es in der agilen Entwicklung grundsätzlich nicht mehr. Sind sie der Meinung, dass das Testing grundsätzlich dem agilen Team selbst überlassen werden soll, oder sollte sich ein Teammitglied dieser Thematik speziell annehmen?

Dies hängt vom Umfang der Test Aktivitäten und der Anzahl der beteiligten Teams ab. Sind mehrere Teams involviert, dann besteht ein erhöhter Bedarf an Koordination. Ab einem gewissen Umfang setze ich jeweils einen Testmanger zu diesem Zweck ein.

2.5 Kennen Sie Unternehmen, welche bei ERP-Einführungsprojekten und/oder im ERP-Betrieb Testautomatisierungen betreiben? Wenn ja, gibt es Ausprägungen/Merkmale bei diesen Unternehmen (Grösse / Branche / spezielle regulatorische Bestimmungen), welche Testautomatisierung begünstigen?

Ja. Grundsätzlich grössere Unternehmen in einem stark reglementierten Bereich.

3. Systemintegration und Schnittstellen

3.1 Ich stelle die These auf, dass bei Schnittstellen mitunter der grösste Testaufwand in einem ERP-Projekt entsteht. Können Sie diese These bestätigen?

Teilweise. Werden bestehende Prozesse 1:1 umgesetzt, dann nehmen die Schnittstellen einen beachtlichen Teil des Testings ein. Aus meiner Sicht ist das Testing von Schnittstellen eher ein einfacherer Teil betreffen Test Case und Ablaufbeschreibung. Werden neue Prozesse implementiert und durch die Software unterstützt, dann gehe ich davon aus, dass die Aufwendungen für das Testing der Schnittstellen einen eher kleineren Teil einnehmen.

3.2 In Scrum werden Stories nach Ihrem Business-Value priorisiert und umgesetzt. Trotz dieses Grundsatzes bin ich der Meinung, dass unterschiedliche Stories welche dieselbe Schnittstelle betreffen, nicht zu unterschiedlichen Zeitpunkten umgesetzt werden sollten und eine Schnittstelle möglichst in ihrer gesamten Ausprägung entwickelt werden sollte. Teilen Sie diese Meinung? Welche Vor- und Nachteile sehen Sie bei dieser Vorgehensweise?

Gute Erfahrungen habe ich mit dem Ansatz, dass der Business Value mit dem Risiko zur Umsetzung kombiniert betrachtet werden sollte. Start mit hohem Nutzen und den höchsten Risiken führt dazu, dass zu einem frühen Zeitpunkt die für das Projekt wichtigsten Erkenntnisse gewonnen werden können. Der Business Value sollte sich über einen gesamten Prozess definieren und nicht durch einzelne Features. Damit steht auch bereits fest, welche Schnittstellen zu dieser Story / zu diesem Release / zu dieser Iteration dazugehören. Diese sind dann entsprechend in den Test zu berücksichtigen.

3.3 *Sollten aus Ihrer Sicht Schnittstellen und dazugehörige Testfälle besonders detailliert beschrieben werden? Auch in agilen Projekten?*

Nein

4. ERP Betrieb

4.1 Diese Frage bezieht sich nicht auf eine spezifische ERP-Software: Wenn auf Seite ERP-Systemanbieter ein neuer Release ausgeliefert wird, bei welchen Komponenten (z.B. Funktionen / Schnittstellen / Individualanpassungen… usw.) sollte aus Ihrer Sicht der Kunde generell ein spezielles Augenmerk bei den Tests legen?

Kann so nicht beantwortet werden.

4.2 Wie ist die ERP-Betriebsorganisation auf Seite des Kunden aus Ihrer Sicht optimalerweise organisiert? Bleibt das agile Team auch nach Projektabschluss in seinen Grundzügen bestehen (z.B. für Changemanagement) oder wird dieses Team im Normalfall aufgelöst und eine separate Betriebsorganisation gebildet?

Aus meiner Erfahrung heraus sind Personen, welche sich in Projekten wohl fühlen und sich dort einbringen wollen mehr interessiert am nächsten Projekt als wiederkehrenden Aufgaben und eher kleineren Changes.

Anhang B

Interview mit ERP-Systemanbieter

Der Interviewpartner möchte, dass Informationen, welche auf Ihn oder sein Unternehmen schliessen lassen, anonymisiert werden. Diesem Wunsch wird nachfolgend Rechnung getragen. Das Original-Interview liegt dem Verfasser vor.

Thema: Agiles Testen in ERP-Projekten

Ziel des Interviews: Durch die Erfahrung von Experten soll das Testen in agilen ERP-Projekten genauer untersucht werden. Welche Vorgehensweisen haben sich aus Sicht ERP-Systemanbieter bewährt und was muss speziell beachtet werden.

Datum: 15.01.2020

Ort: anonymisiert

Art des Interviews: telefonisch und schriftlich

Interviewpartner: Hans Meier (Name geändert)

Funktion: Geschäftsführer und Gesellschafter

Geburtsdatum: anonymisiert

Unternehmen: Ein ERP-Systemanbieter in der DACH Region

Generelles: Die nachfolgenden Fragen sind freiwillig und müssen nicht beantwortet werden. Auf Ihren Wunsch werden Informationen, welche auf Sie oder Ihr Unternehmen schliessen lassen, anonymisiert. Telefonisch- oder persönlich durchgeführte Interviews werden aufgezeichnet, damit sie paraphrasiert und in der Arbeit verwendet werden können. Aufzeichnungen werden nicht aufbewahrt.

Anonymität: Bitte Personen- und Unternehmensinformationen anonymisieren

1. Allgemeine Fragen zu agilen ERP-Projekten

1.1 Sehen Sie in den letzten Jahren eine Tendenz, dass ERP-Projekte vermehrt agil umgesetzt werden? Wenn ja, welche agilen Methoden (z.B. Scrum, Kanban… usw.) werden dabei hauptsächlich angewendet?

Ich denke vermehrt mit Scrum

1.2 Welche **Vorteile** sehen Sie bei einer agilen ERP-Einführung im Vergleich zu herkömmlichen Vorgehensweisen (z.B. Wasserfall)?

Der Kunde sieht eher, was geplant ist und kann Anpassungen/Wünsche einbringen, bevor das «fertige» Produkt da ist und Änderungen aufwändiger werden.

1.3 Welche **Nachteile** sehen Sie bei einer agilen ERP-Einführung im Vergleich zu herkömmlichen Vorgehensweisen (z.B. Wasserfall)?

Weil sich Anforderungen ändern können, sind evtl. auch andere Aufwände zu erwarten. Der Kunde muss wissen, dass ein besser auf ihn zugeschnittenes Produkt dann evtl. etwas teurer wird.

1.4 Wenn wir nun annehmen, dass Scrum die präferierte agile Einführungsmethodik ist, wie würden Sie ein ERP-Scrum-Team aus Ihrer Sicht zusammenstellen? Welche Fähigkeiten sollten die Teammitglieder besitzen?

Können wir nicht beantworten

1.5 Waren bei Ihren (agilen) ERP-Projekten die Mitglieder der agilen Teams über die gesamte Projektdauer dieselben, oder wurden die Teammitglieder je nach Sprint / Themengebiet ausgetauscht?

Bei uns waren es immer dieselben Mitglieder pro Projekt

1.6 Haben Sie Erfahrung in Projekten gemacht, wo mehrere agile Teams an derselben ERP-Einführung mitgearbeitet haben? Wenn ja, aufgrund welcher Kriterien (z.B. ERP-Module, Prozesse, Schnittstellen) wurden die Aufgaben den unterschiedlichen Teams zugewiesen?

Nein, wir haben dazu zu kleine Projekte

2. Fragen zum Testing in agilen ERP Projekten

2.1 Wo / bei welchen Aktivitäten liegt aus Ihrer Sicht der grösste Testaufwand in ERP-Einführungsprojekten?

In Kettentest der Gesamtfunktionalität und der Berücksichtigung möglichst aller wichtigen Geschäftsfallarten

2.2 Wie hoch (Prozentsatz von total aufgewendeter Zeit) schätzen Sie den Aufwand für das Testing bei einer ERP-Einführung?

Schwer zu sagen, da immer wieder an unterschiedlichen Stellen getestet werden muss. 10% vielleicht

2.3 Wie, in welcher Form und zu welchem Zweck empfehlen sie Ihren Kunden die Dokumentation von Testfällen? Können Sie den Einsatz von Testdokumentations-Werkzeugen wie Beispielsweise Jira empfehlen?

Das haben wir bisher meist mit Excel/Word (Sharepoint Online) gemacht, aber wir empfehlen eine Testdokumentation

2.4 Ich stelle die These auf, dass bei Schnittstellen mitunter der grösste Testaufwand in einem ERP-Projekt entsteht. Können Sie diese These bestätigen?

Ja, weil da noch weitere Partner beteiligt sind, ist das immer sehr aufwändig.

2.5 Die Rolle als Testmanager gibt es in der agilen Entwicklung grundsätzlich nicht mehr. Sind sie der Meinung, dass das Testing grundsätzlich dem agilen Team selbst überlassen werden soll, oder sollte sich ein Teammitglied dieser Thematik speziell annehmen?

Ein Projektleiter, der die Anforderungen des Kunden kennt, wird für Tests oft beigezogen.

2.6 Kennen Sie Unternehmen, welche bei ERP-Einführungsprojekten und/oder im ERP-Betrieb Testautomatisierungen betreiben? Wenn ja, gibt es Ausprägungen/Merkmale bei diesen Unternehmen (Grösse / Branche / spezielle regulatorische Bestimmungen), welche Testautomatisierung begünstigen?

Nein, das nutzen und kennen wir nicht.

2.7 Entwickeln Sie als ERP-Systemhersteller Ihre Software agil weiter? Wenn ja, worin liegen aus Ihrer Sicht die grössten Herausforderungen und Stolpersteine?

Ja das tun wir. Die grösste Herausforderung ist, dass wir die Anforderungen aus ganz unterschiedlichen Kanälen erhalten. (Endkunden, Vertriebspartner)

2.8 Führen Sie bei der Weiterentwicklung Ihrer ERP-Software automatisierte Tests durch? Wenn ja, welche Erfahrungen haben Sie dabei gemacht?

Selektiv ja. Der Aufwand für das Schreiben der Tests ist meist hoch, dafür gibt es Sicherheit bei Anpassungen, dass noch alles wie bisher funktioniert.

3. ERP Betrieb

3.1 Diese Frage bezieht sich nicht auf eine spezifische ERP-Software: Wenn auf Seite ERP-Systemanbieter ein neuer Release ausgeliefert wird, bei welchen Komponenten (z.B. Funktionen / Schnittstellen / Individualanpassungen… usw.) sollte aus Ihrer Sicht der Kunde generell ein spezielles Augenmerk bei den Tests legen?

Bei Individualanpassungen

3.2 Wie ist die ERP-Betriebsorganisation auf Seite des Kunden aus Ihrer Sicht optimalerweise organisiert? Bleibt das agile Team auch nach Projektabschluss in seinen Grundzügen bestehen (z.B. für Changemanagement) oder wird dieses Team im Normalfall aufgelöst und eine separate Betriebsorganisation gebildet?

Es wird in der Regel aufgelöst. Das hängt mit unserer Kundengrösse zusammen.

4. Weiteres

4.1 Haben weitere Anmerkungen zu diesem Themengebiet?

Nein, keine weiteren Anmerkungen

Anhang C

Interview mit Agile Coach

Thema:	Agiles Testing in ERP-Projekten
Ziel des Interviews:	Durch die Erfahrung von Experten soll das Testing in agilen ERP-Projekten genauer untersucht werden. Welche Vorgehensweisen haben sich aus Sicht eines Agile Coaches bewährt und was muss speziell beachtet werden.
Datum:	13.01.2020
Ort:	Chur
Art des Interviews:	telefonisch und schriftlich
Interviewpartner:	Klaus Bucka-Lassen
Funktion:	Agile Coach / Managing Director
Geburtsdatum:	**07.07.1969**
Unternehmen:	aragost ag, 8903 Birmensdorf
Generelles:	Die nachfolgenden Fragen sind freiwillig und müssen nicht beantwortet werden. Auf Ihren Wunsch werden Informationen, welche auf Sie oder Ihr Unternehmen schliessen lassen, anonymisiert. Telefonisch- oder persönlich durchgeführte Interviews werden aufgezeichnet, damit sie paraphrasiert und in der Arbeit verwendet werden können. Aufzeichnungen werden nicht aufbewahrt.
Anonymität:	Personen- und Unternehmensinformationen dürfen in der Arbeit vermerkt werden

1. Allgemeine Fragen zu agilen Projekten

1.1 Sehen Sie in den letzten Jahren eine Tendenz, dass auch Projekte, welche nicht der klassischen Softwareentwicklung zugeordnet werden können, vermehrt agil umgesetzt werden? Wenn Ja, welche agilen Methoden (z.B. Scrum, Kanban… usw.) werden dabei hauptsächlich angewendet?

Ja, definitiv. Agilität ist überall dort notwendig wo die Komplexität zunimmt und das ist in praktisch allen Bereichen der Fall. Scrum ist ja eigentlich keine Methode, sondern ein Rahmenwerk, ein Gerüst, bei dem man die Methode wie man tatsächlich vorgeht selber definiert. Ähnlich mit Kanban. Beide diese Vorgehensweisen werden oft auch ausserhalb der Softwareentwicklung angewandt.

1.2 Welche **Vorteile** sehen Sie bei einer agilen ERP-Einführung im Vergleich zu herkömmlichen Vorgehensweisen (z.B. Wasserfall)?

Nun habe ich nicht viel Erfahrung mit ERP Einführungen, aber ich nehme schwer an, dass sich bei diesen genau dieselben Probleme zeigen wie bei allen anderen Software-Einführungen. Zu Beginn eines solchen Projektes weiss man per Definition am wenigsten. Mit jedem Tag wird man schlauer, weiss man mehr was benötigt wird, versteht man den Kontext in dem das ERP System funktionieren soll besser. Und trotzdem trifft man so viele Entscheidungen wie möglich ganz am Anfang eines solchen Projektes – also zu dem Zeitpunkt an dem man am wenigsten weiss. Eines der Prinzipien aus Lean besagt, dass man Entscheidungen so spät wie möglich und so früh wie nötig treffen soll. Das ist auch Agilität und erlaubt es bessere Entscheide zu treffen. Ausserdem kann man viel sparen, wenn man nicht unnötig Energie in eine frühzeitige Entscheidungsfindung investiert.

1.3 Welche **Nachteile** sehen Sie bei einer agilen ERP-Einführung im Vergleich zu herkömmlichen Vorgehensweisen (z.B. Wasserfall)?

Höchstens, dass es eine neue Vorgehensweise und deshalb ausserhalb der Komfortzone der meisten Personen ist.

1.4 Wenn wir nun annehmen, dass Scrum die präferierte agile ERP-Einführungsmethodik ist, wie würden Sie ein ERP-Scrum-Team aus Ihrer Sicht zusammenstellen? Welche Fähigkeiten sollten die Teammitglieder aus Ihrer Sicht besitzen?

Das Team als Ganzes sollte Interdisziplinär sein und zwar so weit, dass das Team alles machen kann was es braucht, damit eine Anforderung in produktive Funktionalität umgesetzt werden kann.

1.5 ERP-Projekte sind bekanntlich stark auf die Unternehmensprozesse fokussiert und im Vergleich zur klassischen Softwareentwicklung wird wenig «programmiert». Entsprechend sind Prozessexperten und Fachspezialisten wichtige Informationsträger in diesem Projekt. Wie würden Sie diese Personen im Projekt einbinden? Sind sie bevorzugt Mitglieder des Scrum-Teams oder lediglich als Stakeholder zu betrachten?

Als Stakeholder würde ich sie wahrscheinlich nicht sehen. Es ist wichtig, dass das Team direkt mit den späteren Anwendern des ERP Systems (die wichtigsten Stakeholder) sprechen kann und auch tut. Es braucht keine Leute die das «übersetzen». Ich würde sie also definitiv bevorzugt als 100% Mitglieder eines Teams sehen.

1.6 Wenn in einem agilen ERP-Projekt mehrere Scrum-Teams mitarbeiten, aufgrund welcher Kriterien würden Sie Features und Stories den einzelnen Teams zuweisen? Ist es korrekt, dass bei der Aufteilung der Stories darauf geachtet werden soll, dass zwischen den Teams möglichst wenig Berührungspunkte entstehen? Welche Überlegungen müssten hier vorgängig gemacht werden?

Ja, so wenig Berührungspunkte zwischen den Teams wie möglich, aber natürlich die, die nötig sind (siehe auch oben). Es gilt also nicht die Teams zu skalieren, sondern darum das Produkt (wir bevorzugen von Produkten und nicht Projekten zu reden) zu «deskalieren», also so in Module oder Teilprodukte zu stückeln, dass einzelne Teams mehr-oder-weniger unabhängig an ihrem Modul arbeiten können. Ich würde diesen Teams keine Arbeit zuweisen, sondern sie selber die Arbeiten ziehen lassen, für die sie sich fit sehen. Das nennt man Pull und das funktioniert meistens wesentlich effizienter als Push (= zuweisen), weil Push eine grosse Gefahr für Stau birgt (ähnlich wie auf einer Autobahn).

2. Fragen zum Testing in agilen Projekten

2.1 Wie, in welcher Form und zu welchem Zweck empfehlen sie die Dokumentation von Testfällen? Können Sie den Einsatz von Testdokumentations-Werkzeugen wie Beispielsweise Jira empfehlen?

Das kann ich so nicht beantworten

2.2 Die Rolle als Testmanager gibt es in der agilen Entwicklung grundsätzlich nicht mehr. Sind sie der Meinung, dass das Testing grundsätzlich dem agilen Team selbst überlassen werden soll, oder sollte sich ein Teammitglied dieser Thematik speziell annehmen?

Es wird definitiv dem Team überlassen – wie dieses das dann intern löst ist dem Team überlassen. Sollten sie sich dafür entscheiden tatsächlich eine Rolle dafür zu definieren ist das ihre Sache, nicht die eines Managers ausserhalb des Teams.

2.3 Wenn wir nun annehmen, dass ein spezifisches Scrum-Teammitglied die Rolle als Tester/in übernimmt, welches sind die Erfolgsfaktoren eines guten Testers? Wo sollten Tester/innen einen speziellen Fokus legen?

Was oft missverstanden wird ist, dass es die Aufgabe eines Testers ist zu beweisen, dass ein System funktioniert. Das ist Quatsch. Das geht gar nicht. Der Job eines Testers ist es zu beweisen, dass ein System nicht funktioniert. Wenn man das so sieht hat die Konsequenzen für das ganze Team. Die Mentalität ist nicht mehr «Ich hacke mal etwas zusammen und werfe es dann über den Zaun zu den Testern – die müssen uns dann sagen was wir fixen müssen».

2.4 In der agilen Softwareentwicklung wird Testautomatisierung aufgrund der Literatur nahezu vorausgesetzt. Haben Sie selbst Erfahrungen mit Testautomatisierung gemacht? Wenn ja welche?

Leider viel zu wenig. Testautomatisierung steckt bei vielen traditionellen Unternehmen (Banken, Versicherungen, etc.) noch in den Kinderschuhen.

2.5 Sehen Sie bei Softwareprojekten, bei welchen tendenziell wenig programmiert wird, Testautomatisierungswerkzeuge als ein geeignetes Werkzeug?

Das kann ich so nicht beantworten

2.6 Können Sie bezüglich Einführung und Anwendung von Testautomatisierungs-Werkzeugen eine Empfehlung abgeben (Tipps und Tricks)?

Klein anfangen und Schritt für Schritt erweitern. Nicht – wie so oft – Wasserfallmässig anrühren mit grossen Analysen (welches ist das beste Tool), Designs und Plänen wie man Testautomatisierung einführt.

3. ERP Betrieb

3.1 Wie ist die ERP-Betriebsorganisation auf Seite des Kunden aus Ihrer Sicht optimalerweise organisiert? Bleibt das agile Team auch nach Projektabschluss in seinen Grundzügen bestehen (z.B. für Changemanagement) oder wird dieses Team im Normalfall aufgelöst und eine separate Betriebsorganisation gebildet?

Es bleibt bestehen – siehe Produkt vs. Projekt Diskussion. Es bekommt (bzw. zieht sich) dann evtl. noch weitere Aufgaben / Produkte.

4. Weiteres

4.1 Haben weitere Anmerkungen?

Keine weiteren Anmerkungen

Anhang D

Interview mit unabhängigem ERP- Projektleiter

Thema:	Agiles Testing in ERP-Projekten
Ziel des Interviews:	Durch die Erfahrung von Experten soll das Testing in agilen ERP-Projekten genauer untersucht werden. Welche Vorgehensweisen haben sich aus Sicht ERP-Projektleiter bewährt und was muss speziell beachtet werden.
Datum:	13.01.2020
Ort:	**Scuol, 16.01.2020**
Art des Interviews:	schriftlich / persönlich
Interviewpartner:	Reto Hansmann
Funktion:	IT-Projektleiter
Geburtsdatum:	**16. Oktober 1962**
Unternehmen:	Rhätische Bahn AG, 7000 Chur
Generelles:	Die nachfolgenden Fragen sind freiwillig und müssen nicht beantwortet werden. Auf Ihren Wunsch werden Informationen, welche auf Sie oder Ihr Unternehmen schliessen lassen, anonymisiert. Telefonisch- oder persönlich durchgeführte Interviews werden aufgezeichnet, damit sie paraphrasiert und in der Arbeit verwendet werden können. Aufzeichnungen werden nicht aufbewahrt.
Anonymität:	Personen- und Unternehmensinformationen dürfen in der Arbeit vermerkt werden

1. Allgemeine Fragen zu ERP-Projekten

1.1 Sehen Sie in den letzten Jahren eine Tendenz, dass ERP-Projekte vermehrt agil umgesetzt werden? Wenn Ja, welche agilen Methoden (z.B. Scrum, Kanban… usw.) werden dabei hauptsächlich angewendet?

Aufgrund des Umfangs einer Gesamteinführung von ERP Projekten ist es schwierig, nur auf agile Methoden zu setzen. Solche Projekte laufen nach wie vor meist nach

herkömmlichen Projektmethoden (PMI, IPMA, Hermes usw.) ab. Diese werden aber häufig kombiniert mit agilen Methoden in einzelnen Bereichen und/oder Teilprojekten. Dabei wird meist Scrum - wenn auch nicht immer 'lupengenau' - angewendet.

1.2 Welche **Vorteile** sehen Sie bei einer agilen ERP-Einführung im Vergleich zu herkömmlichen Vorgehensweisen (z.B. Wasserfall)?

- Hohe Flexibilität/Agilität durch adaptives Planen
- Kurze Kommunikationswege
- Hohe Transparenz durch regelmässige Meetings
- Zeitnahe Realisation neuer Produkteigenschaften bzw. Inkremente
- Kontinuierlicher Verbesserungsprozess
- Kurzfristige Problem-Identifikation
- Geringer Administrations- und Dokumentationsaufwand

1.3 Welche **Nachteile** sehen Sie bei einer agilen ERP-Einführung im Vergleich zu herkömmlichen Vorgehensweisen (z.B. Wasserfall)?

- *Hoher Kommunikations- und Abstimmungsaufwand*
- *Kein Gesamtüberblick über die komplette Projektstrecke*
- *Wenige konkrete Handlungsempfehlungen*
- *Zeitverluste bei zu „defensiven" Sprintplanungen*
- *„Tunnelblick-Gefahr" bei ausschließlicher Fokussierung auf Tasks*
- *Erschwerte Koordination mehrerer Entwicklungs-Teams bei Grossprojekten*
- *Potenzielle Verunsicherung aufgrund fehlender Zuständigkeiten und Hierarchien*
- *Potenzielle Unvereinbarkeit mit bestehenden Unternehmensstrukturen*

2. Fragen zum Testing in (agilen) ERP Projekten

2.1 Wo / bei welchen Aktivitäten liegt aus Ihrer Sicht der grösste Testaufwand in ERP-Einführungsprojekten?

Beim Testen von:

- *Wertefluss*
- *Gesamtintegration*
- *Schnittstellen*
- *Abschlussarbeiten wie z. B. Jahresendverarbeitung*

2.2 Wie hoch (Prozentsatz von total aufgewendeter Zeit) schätzen Sie den Aufwand für das Testing bei einer ERP-Einführung?

Ich schätze ca. 25 – 35%

2.3 Wie, in welcher Form und zu welchem Zweck empfehlen sie den die Dokumentation von Testfällen im laufenden Projekt? Können Sie den Einsatz von Testdokumentations-Werkzeugen wie Beispielsweise Jira empfehlen?

JA unbedingt. Die Testfälle müssen standardisiert sein, damit sie mehrfach ausgeführt werden können. Tools können helfen, sind aber m. E. nicht unbedingt nötig. Ein Set an (gleichbleibenden) Grunddaten mit welchen die Test-Szenarien durchgespielt werden können, erachte ich eher als ein Muss.

Wichtig ist die Testresultate fest zu halten, ob manuell oder mittels Toolunterstützung ist m. E. nicht relevant.

2.4 Wenn das ERP-System erstmals eingeführt wurde, werden vom ERP-Systemlieferanten stetig neue Releases zur Verfügung gestellt. Sollen aus Ihrer Sicht die Testfälle aus dem ERP-Einführungsprojekt wiederverwendet werden? Wenn ja, welche?

Ja unbedingt.

Meistens können nach einem Releasewechsel nicht wirklich alle Funktionen ausgetestet werden (z.B. Jahresabschluss). Wichtig ist jedoch, dass vom normalen Tagesgeschäft die Funktionen getestet werden. Dies erspart Überraschungen im Tagesgeschäft.

2.5 Diese Frage bezieht sich nicht auf eine spezifische ERP-Software: Wenn auf Seite ERP-Systemanbieter ein neuer Release ausgeliefert wird, bei welchen Komponenten (z.B. Funktionen / Schnittstellen / Individualanpassungen… usw.) sollte aus Ihrer Sicht der Kunde generell ein spezielles Augenmerk bei den Tests legen?

-Tagesgeschäft (wie oben erwähnt)
- Integration in die betriebsinterne Infrastruktur (sofern vorhanden) und Schnittstellen

2.6 Die Rolle als Testmanager gibt es in der agilen Entwicklung grundsätzlich nicht mehr. Sind sie der Meinung, dass das Testing grundsätzlich dem agilen Team selbst überlassen werden soll, oder sollte sich ein Teammitglied dieser Thematik speziell annehmen?

M. E. sollte immer eine Person für das Gesamtresultat verantwortlich sein. Das gilt auch für die Tests.

2.7 Kennen Sie Unternehmen, welche bei ERP-Einführungsprojekten und/oder im ERP-Betrieb Testautomatisierungen betreiben? Wenn ja, gibt es Ausprägungen/Merkmale bei diesen Unternehmen (Grösse / Branche / spezielle regulatorische Bestimmungen), welche Testautomatisierung begünstigen?

Das kann ich so nicht beantworten, mir fehlt dazu die Übersicht.

3. Systemintegration und Schnittstellen

3.1 Ich stelle die These auf, dass bei Schnittstellen mitunter der grösste Testaufwand in einem ERP-Projekt entsteht. Können Sie diese These bestätigen?

Jein…

M. E. ist der Wertefluss am entschiedensten und dieser gibt insgesamt auch den meisten Aufwand.

Um den Wertefluss abbilden und testen zu können, sind aber unter Anderem Schnittstellen relevant externe und/oder Modulübergreifend. Von da her ja.

3.2 Was muss bei Schnittstellentests generell beachtet werden? Gibt es Techniken, welche die Tests von Schnittstellen zusätzlich unterstützen?

Das kann ich nicht pauschal beantworten.

3.3 Sollten aus Ihrer Sicht Schnittstellen und dazugehörige Testfälle besonders detailliert beschrieben werden? Auch in agilen Projekten?

Nein, normal wie alles Andere.

3.4 Haben Sie weitere Tipps bezüglich der Entwicklung und den Test von Schnittstellen in einem ERP-Projekt?

Wenn immer möglich Standards benutzen.

4. ERP Betrieb

4.1 Wie ist die ERP-Betriebsorganisation auf Seite des Kunden aus Ihrer Sicht optimalerweise organisiert? Bleibt das agile Team auch nach Projektabschluss in seinen Grundzügen bestehen (z.B. für Changemanagement) oder wird dieses Team im Normalfall aufgelöst und eine separate Betriebsorganisation gebildet?

Der Betrieb sollte nicht agil organisiert sein und wenn möglich nach ITIL Grundsätzen erfolgen. Falls die Betriebsgrösse es zulässt auch ein agiles Team nur Weiterentwicklung und Changemanagement zu unterhalten, ist das zu empfehlen.

5. Weiteres

5.1 Würden Sie generell eine agile Umsetzung eines ERP-Projektes empfehlen, oder tendieren Sie weiterhin auf konventionelle jedoch bewährte ERP-Einführungsmethoden zurückzugreifen? Wieso?

M E ist eine Mischfunktion am idealsten, Konventionelle Wasserfallmethode, kombiniert mit Agilität bei der Entwicklung/Test.

5.2 Haben weitere Anmerkungen?

Testfälle und erwartete Resultate so früh wie möglich festlegen! Nicht erst beim Testen.

Nutzungs-/Verwendungsrechte an der Masterarbeit

Solange nichts anderes vereinbart ist, liegen die Nutzungs- bzw. Verwendungsrechte an der Masterarbeit bei der FHS St.Gallen. Dazu gehört unter anderem das Recht, die Masterarbeit zu publizieren. Die Verfasser dürfen Masterarbeiten nur mit Zustimmung der Weiterbildungsleitung veröffentlichen oder auf andere Weise verwerten. Zur Einholung der Zustimmung haben sie ein **schriftliches, begründetes Gesuch bei der zuständigen Studienkoordinatorin einzureichen.**

Die Verfasser können bei der Weiterbildungsleitung beantragen, dass die Masterarbeit vertraulich behandelt wird, wenn deren Nutzung bzw. Verwendung durch die FHS St.Gallen Berufs- bzw. Geschäftsgeheimnisse von Dritten verletzen oder Persönlichkeitsrechte von Dritten tangieren würde. **Der schriftliche Antrag ist bei der zuständigen Studienkoordinatorin einzureichen.** Die behauptete Verletzung von Berufs- bzw. Geschäftsgeheimnissen oder das Tangieren von Persönlichkeitsrechten ist im Antrag glaubhaft zu machen.

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit ohne fremde Hilfe und nur unter Benützung der angegebenen Quellen verfasst habe, und dass ich ohne schriftliche Zustimmung der Weiterbildungsleitung der FHS St.Gallen keine Kopien dieser Arbeit an Dritte aushändigen werde.

	Datum	Unterschrift
Nando Costantino	_____	_____