Thomas Purkarthofer, BSc

Agile Softwareentwicklung in sicherheitskritischen Bereichen wie Automobil- und Medizintechnik

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Biomedical Engineering

submitted to

Graz University of Technology

Supervisor

Assoc.Prof. Dipl.-Ing. Dr.techn. Jörg Schröttner

Institut of Health Care Engineering mit Europaprüfstelle für Medizinprodukte

DI Ivana Stojanovic (AVL List GmbH)

Graz, September 2018

Eidesstattliche Erklärung
Affidavit

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis/diploma thesis/doctoral dissertation.

Datum/Date	Unterschrift/Signature

Die Technische Universität Graz übernimmt mit der Betreuung und Bewertung einer Masterarbeit keine Haftung für die erarbeiteten Ergebnisse: Eine positive Bewertung und Anerkennung (Approbation) einer Arbeit bescheinigt nicht notwendigerweise die vollständige Richtigkeit der Ergebnisse.

Danksagung

Als Erstes gilt mein Dank Frau DI Ivana Stojanovic, die mich von seitens der AVL List GmbH betreut hat und ohne ihre Unterstützung die Anfertigung dieser Masterarbeit nicht möglich gewesen wäre.

Ebenfalls möchte ich mich bei meinem Betreuer vom Institut für Health Care Engineering an der Technischen Universität Graz, Herrn Assoc.Prof. Dipl.-Ing. Dr.techn. Jörg Schröttner recht herzlich bedanken, der mir die Durchführung der Masterarbeit ermöglicht hat, mir stets hilfreiche Ratschläge erteilt und meine Arbeit begutachtet hat.

Ein besonderer Dank gilt auch Frau DI Tanja Vogrin und Herrn DI Otto-Wilhelm Herschmann, die mir die Durchführung der Masterarbeit in der AVL List GmbH ermöglichen konnten.

Zum Abschluss möchte ich mich noch bei all meinen Freunden und meiner Familie für den emotionallen Rückhalt und die Unterstützung während meines gesamten Studiums bedanken.

Kurzfassung

Agile Softwareentwicklung in sicherheitskritischen Bereichen wie Automobil- und Medizintechnik: In dieser Arbeit wurde eine mögliche Anwendung der beiden agilen Softwareentwicklungsmethoden Scrum und Kanban im Automobil- und Medizintechnikbereich näher analysiert. Ein Vergleich zwischen den Standards EN 62304 und Automotive SPICE 3.1 auf inhaltliche Übereinstimmung wurde durchgeführt und mit den daraus gewonnenen Erkenntnissen die beiden agilen Methoden Scrum und Kanban auf mögliche Anwendbarkeit in ausgewählten Aktivitäten und Prozessen analysiert. Um etwaige Lücken zwischen Scrum/Kanban und den geforderden Standards auszugleichen und aufzuzeigen, dass Anforderungen aus den Normen mit agilen Methoden erfüllt werden können, wurde im Zuge dieser Masterabeit ein Lösungsvorschlag in Form eines Scrum-Lebenszyklus erstellt. Daraus ergaben sich folgende Resultate: Der Vergleich zwischen den Prozessen aus Automotive SPICE und den Aktivitäten aus EN 62304 lieferte sehr unterschiedliche Ergebnisse, welche von vollständiger bis hin zur fehlenden Übereinstimmung der Prozess-Aktivitätenpaare reichten. Weiters wurde durch die durchgeführten Gap-Analysen festgestellt, dass eine ausnahmslose Anwendung von Scrum und Kanban bei den ausgewählten Prozessen/Aktivitäten nicht normkonform wäre. Durch die Zuhilfenahme von zusätzlichen Softwareentwicklungspraktiken könnte ein Scrum-Lebenszyklus, welcher alle Anforderungen der Normen erfüllen würde, entwickelt werden. Schlüsselwörter: *Agile Softwareentwicklung, EN 62304, Automotive SPICE 3.1, Scrum, Kanban*

Abstract

Agile software development in safety critical areas such as automotive and medical engineering: For this study a possible application of agile software development methods Scrum and Kanban in automotive and medical engineering was analyzed in more detail. First of all a consistency comparison of the content between medical standard EN 62304 and automotive standard Automotive SPICE 3.1 has been carried out. Based on gained knowledge of partly consistency, Scrum and Kanban could be analyzed on possible usage on chosen activities and processes. In order to compensate gaps between Scrum/Kanban and the required standards, and to demonstrate that requirements from these standards can be fulfilled with agile methods, a Scrum lifecycle as proposed solution was created. The results of this study are as follows: Comparison between processes (Automotive SPICE) and activities (EN 62304) has shown very different results, ranging from complete to lack of conformity of the process-activity pairs. An additional conclusion of gap analyses was that the compatibility of Scrum and Kanban with these standards is only possible with further non-agile methods. In order to that a standard compatible Scrum Lifecycle could be developed. Keywords: *Agile software development, EN 62304, Automotive SPICE 3.1, Scrum, Kanban*

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Motivation . . . 1
1.2 Agile Softwareentwicklung . . . 2
1.2.1 Agiles Manifest . . . 3
1.2.2 Prinzipien der agilen SW Entwicklung . . . 5
1.2.3 Einsatzbereich von agiler Softwareentwicklung . . . 7
1.3 Agile Methoden . . . 8
1.3.1 Scrum . . . 10
1.3.1.1 Lebenszyklus . . . 10
1.3.1.2 Rollen . . . 11
1.3.1.3 Ereignisse . . . 12
1.3.1.4 Artefakte . . . 14
1.3.2 Kanban . . . 17
1.3.2.1 Die Prinzipien von Kanban: . . . 17
1.3.2.2 Hauptpraktiken von Kanban . . . 17
1.4 Harmonisierte Normen . . . 21
1.4.1 Automotive SPICE - Process Assessment/Reference Model 3.1 . . . 21
1.4.1.1 Struktur des Prozessreferenzmodells (PRM) . . . 22
1.4.2 Medizingeräte-Software - Software-Lebenszyklus-Prozesse (EN 62304: 2006) . . . 24
1.4.2.1 Lebenszyklus der Softwareentwicklung von Medizinprodukten 25

2 Aufgabenstellung **27**

3 Methoden **29**
3.1 EN 62304 und Automotive SPICE 3.1 im Vergleich . . . 29
3.2 Übereinstimmungen und Konflikte von ausgewählten Prozessen bzw. Aktivitäten mit agiler Softwareentwicklung . . . 31
3.3 Entwicklung eines agilen Scrum- Lebenszyklus für ausgewählte Prozesse/Aktivitäten . . . 32

4 Ergebnisse **35**
4.1 EN62304 und Automotive SPICE 3.1 im Vergleich . . . 35

4.2 Übereinstimmungen und Konflikte von ausgewählten Prozessen bzw. Aktivitäten mit agiler Softwareentwicklung 36
4.2.1 Herausforderungen bei der agilen Analyse von Softwareanforderungen 36
4.2.1.1 Softwareanforderungsanalyse (SWE.1) 37
4.2.1.2 Analyse der Softwareanforderungen (§ 5.2) 37
4.2.2 Herausforderungen bei der agilen Analyse von Änderungen 41
4.2.2.1 Änderungsantragsmanagement (SUP.10) 41
4.2.2.2 Änderungskontrolle (§ 8.2) 41
4.2.3 Herausforderungen bei der agilen Analyse von Problemen 44
4.2.3.1 Problemlösungsmanagement (SUP.9) 44
4.2.3.2 Untersuchung des Problems (§ 9.2) 44
4.3 Entwicklung eines agilen Scrum-Lebenszyklus für ausgewählte Prozesse/Aktivitäten 47
4.3.1 Analyse von Softwareanforderungen (SWE.1/§ 5.2) - Lösungsvorschlag zur agilen Umsetzung der Anforderungen aus den Normen . 47
4.3.2 Analyse von Änderungen (SUP.10/§ 8.2) - Lösungsvorschlag zur agilen Umsetzung der Anforderungen aus den Normen 49
4.3.3 Analyse von Problemen (SUP.9/§ 9.2) - Lösungsvorschlag zur agilen Umsetzung der Anforderungen aus den Normen 50
4.3.4 Scrum-Lebenszyklus zur Erfüllung der Anforderungen aus den ausgewählten Aktivitäten aus EN 62304 und Prozessen aus ASPICE 3.1 51

5 Diskussion **57**
5.1 EN 62304 und Automotive SPICE 3.1 im Vergleich 57
5.2 Übereinstimmungen und Konflikte der ausgewählten Prozesse/Akitiväten mit agiler Softwareentwicklung 58
5.3 Entwicklung eines agilen Scrum- Lebenszyklus für ausgewählte Prozesse/Aktivitäten 59
5.4 Agile Softwareentwicklung und V-Modell im Vergleich 60

6 Schlussfolgerung **61**

7 Anhang **67**

Abbildungsverzeichnis

1.1 „Agile umbrella“ [1] . . . 3
1.2 „Stacey complexity model“ [2] . . . 8
1.3 Agile Methoden und Praktiken - Kugler Maag CIE Umfrage [3] . . . 9
1.4 Agile Methoden - VersionOne Inc. Umfrage [4] . . . 9
1.5 Allgemeiner Scrum-Lebenszyklus [5] . . . 11
1.6 Definition of Done/Definition of Ready- Checklisten [6] . . . 16
1.7 „One day in Kanban land“ [7] . . . 20
1.8 Beziehung zwischen PAM, PRM aus Automotive SPICE (ASPICE) 3.1 [8] und dem Bewertungsrahmen aus ISO/IEC 33020:2015 [9] . . . 22
1.9 ASPICE 3.1 Prozessreferenzmodell im Überblick [8] . . . 23
1.10 ASPICE 3.1 - VDA-Scope der HIS [8] . . . 23
1.11 ASPICE 3.1 Prozessbeschreibung [8] . . . 24
1.12 Softwareentwicklungsprozesse und Aktivitäten unter EN 62304:2006 . . . 25
1.13 V-Modell für die Entwicklung von Medizingerätesoftware laut EN 62304 [10] 26
4.1 Lösungsvorschlag zur agilen normkonformen Umsetzung der ausgewählten Prozesse/Aktivitäten (Lebenszyklus - Teil 1) . . . 54
4.2 Lösungsvorschlag zur agilen normkonformen Umsetzung der ausgewählten Prozesse/Aktivitäten (Lebenszyklus - Teil 2) . . . 55

Tabellenverzeichnis

1 Vergleich zwischen EN 62304 § 5.2 und ASPICE 3.1 Prozess SWE.1 30
2 Prozess-/Aktivitätenvergleich zwischen EN 62304 und ASPICE 3.1 36
3 Gap-Analyse zwischen ASPICE 3.1 - SWE.1 und Scrum/Kanban 39
4 Gap-Analyse zwischen EN 62304 - § 5.2 und Scrum/Kanban 40
5 Gap-Analyse zwischen ASPICE 3.1 - SUP.10 und Scrum/Kanban 42
6 Gap-Analyse zwischen EN 62304 - § 8.2 und Scrum/Kanban 43
7 Gap-Analyse zwischen ASPICE 3.1 - SUP.9 und Scrum/Kanban 45
8 Gap-Analyse zwischen EN 62304 - § 9.2 und Scrum/Kanban 46
9 SWE.1/§ 5.2 konformer Anwendungsvorschlag von Scrum und Kanban . . 48
10 SUP.10/§ 8.2 konformer Anwendungsvorschlag von Scrum und Kanban . . 49
11 SUP.9/§ 9.2 konformer Anwendungsvorschlag von Scrum und Kanban . . . 51

Abkürzungsverzeichnis

ASPICE Automotive SPICE

EN 62304 EN 62304:2006

XP Extreme Programming

DoD Definition of Done

DoR Definition of Ready

PRM Prozessreferenzmodell

PAM Prozessassessmentmodell

HIS Herstellerinitiative-Software

BP Base Practice

VDA Verband der Automobilindustrie

JIRA JIRA Software

1 Einleitung

1.1 Motivation

Flexibilität, Kundennähe und kurze Entwicklungszyklen sind Schlagworte, die in den letzten Jahren im Bereich der Softwareentwicklung immer mehr an Bedeutung gewonnen haben.

Dies ist mit den klassisch verwendeten Softwareentwicklungsmodellen, wie dem Wasserfall- oder dem V-Modell, wie es in vielen konventionellen Industriebereichen für die Entwicklung von Software und Control Systemen verwendet wird, schwer umsetzbar.

Als Beispiel für die unzureichende Flexibilität im V-Modell, kann das Änderungsmanagement angeführt werden. Im Falle einer teilweisen Änderung der Software, wird der Prozess mit der Abänderung der Anforderungen erneut gestartet. Wodurch bereits getätigte Schritte, wie z.B. das Testen der restlichen Software, erneut durchgeführt werden müssen, und somit die Komplexität und der zeitliche Aufwand der Softwareänderung erhöht werden würde.

Weiters ist es in der Softwareentwicklung Standard, dass sich die Halbwertszeit der Anforderungen in den letzten Jahren deutlich verringert hat. Wenn nun beispielsweise die Halbwertszeit für die Anforderungen der Software nur 6 Monate beträgt und eine durschnittliche Projektdauer von 2 Jahren angenommen wird, sind viele dieser Anforderungen schon im ersten Drittel des Projektes wieder veraltet. [11]

Zusätzlich sei noch erwähnt, dass bei Einhaltung des V-Modells eine beinahe monatliche Auslieferung von neuen Softwarepaketen, wie von vielen namhaften Firmen praktiziert, nicht erreicht werden kann.

Aufgrund solcher Anforderungen an die Entwicklung von Software, die mit agilen Methoden leichter umzusetzen wären, geht der Trend immer mehr in diese Richtung. Um einerseits dem Trend der Softwarebranche zu folgen und andererseits, die noch viel wichtigeren firmen- bzw. branchenspezifischen Bedürfnisse zu erfüllen, wurden im Zuge dieser Masterarbeit agile Methoden (Scrum, Kanban) auf ihre Anwendbarkeit in sicherheitskritischen Industriebereichen wie der Automobil- und Medizintechnik überprüft.

Dabei wurden die aufgrund der Einhaltung von harmonisierten Normen resultierenden Problemfelder aufgedeckt, analysiert und Vorschläge zur Umsetzung von agiler Softwareentwicklung in der Automobil- und Medizintechnik entwickelt.

Diese Arbeit ist in sechs Kapitel gegliedert. Das erste Kapitel beschreibt die Motivation der vorliegenden Masterarbeit und Grundsätzliches über agile Softwareentwicklung und harmonisierte Standards aus der Automobil- bzw. Medizintechnik. Der darauffolgende Abschnitt beschäftigt sich mit der Aufschlüsselung der Forschungsfragen, um die Anwendbarkeit von agiler Softwareentwicklung in den beiden Branchen zu überprüfen. Kapitel drei gibt Auskunft über die Gewinnung von Herausforderungen, Problemen und Erfahrungen von agiler Softwareentwicklung in den analysierten Branchen und dem entwickelten Lösungsvorschlag, welche im folgenden Kapitel vier präsentiert und im vorletzten Abschnitt diskutiert werden. Das sechste Kapitel vervollständigt diese Masterarbeit mit abschließenden Bemerkungen.

1.2 Agile Softwareentwicklung

Agile Softwareentwicklungs-Methoden sind erstmals in den späten 1990er aufgetaucht [12], wobei der Ausdruck „agil“ nicht eindeutig definiert ist oder eine spezifische Methode beschreibt, sondern vielmehr einen Regenschirm für Methoden, Praktiken und Prinzipien darstellt (Abb. 1.1). Agile Methoden und Praktiken die unter diesen Regenschirm fallen sind u.a. Scrum [13], Extreme Programming (XP) [14], Kanban oder Continuous Integration. Die Methoden Scrum, Kanban und XP sind die am meisten verbreitetsten agile Methoden der Welt. Dabei fokussiert sich XP auf Entwicklungspraktiken, um regelmäßig und schnell Softwarepakete zu liefern. Auf die Methoden Scrum und Kanban, bzw. Praktiken wie Burn Down Charts oder User Stories wird im Abschnitt 1.3 auf Seite 8 genauer eingegangen.

Aber wodurch zeichnen sich agile Methoden nun eigentlich aus? Agile Methoden sind [15]...

- inkrementell: Das bedeutet, dass es zu relativ kleinen Software-Releases und zu kurzen Zyklen (Scrum: 2-3 wöchige Sprints) kommt. Dadurch kann man auf Fehler sofort reagieren und daraus im nächsten Release lernen. Weiters werden Funktionalitäten auf mehrere Teile und somit auch Sprints aufgeteilt, die inkrementell ausgeliefert werden und die Funktionalität mit jedem Sprint anwachsen lässt. Dabei ist jeder ausgelieferte Teil funktionsfähig.

- kooperativ: Bei agilen Methoden sollte die Kommunikation zwischen Entwicklern und Stakeholder ständig im Vordergrund stehen, und dementsprechend eine konstante

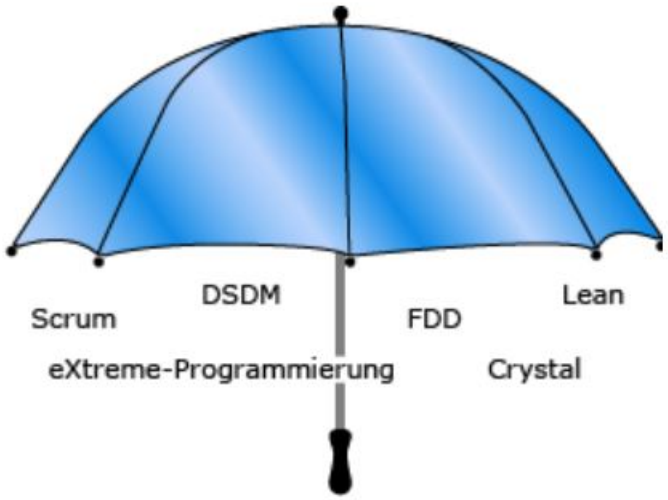


Abb. 1.1: „Agile umbrella“: Agile Methoden und Praktiken wie z.B. Scrum oder XP [1]

Zusammenarbeit gewährleistet werden. Dadurch bekommt der Kunde eine Software die er zum Zeitpunkt der Auslieferung benötigt und nicht die er am Beginn des Projektes in Auftrag gegeben hat.

- geradlinig: Der Begriff „geradlinig“ bedeutet, dass die verwendete Methode gut dokumentiert, einfach zu lernen und zu modifizieren ist.

- adaptiv: Eine weitere Kerneigenschaft der agilen Entwicklung ist die Möglichkeit Änderungen der Anforderungen, sogenannte „changes“, während eines laufenden Projektes einfach zu integrieren. Wodurch es bei Veränderung nicht, wie beim klassischen V-Modell, zu massiven Zeiteinbußen kommen muss. Zusätzlich dazu kann nach jedem Sprint der Arbeitsprozess abgeändert werden, wodurch eventuelle Verbesserungen im Prozess sofort umgesetzt werden könnten.

1.2.1 Agiles Manifest

Der Grundstein der heutigen agilen Softwareentwicklung wurde von einer Gruppe aus 17 Softwareentwicklern bzw. Beratern Anfang 2001 gelegt. Dabei wurde über neue agile Softwareentwicklungsmethoden diskutiert und im Zuge dessen ein Manifest für agile Softwareentwicklung veröffentlicht.

> *„Wir erschließen bessere Wege, Software zu entwickeln, indem wir es selbst tun und anderen dabei helfen. Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:*
>
> - ***Individuen und Interaktionen*** *mehr als Prozesse und Werkzeuge*
> - ***Funktionierende Software*** *mehr als umfassende Dokumentation*
> - ***Zusammenarbeit mit dem Kunden*** *mehr als Vertragsverhandlung*
> - ***Reagieren auf Veränderung*** *mehr als das Befolgen eines Plans*
>
> *Das heißt, obwohl wir die Werte auf der rechten Seite wichtig finden, schätzen wir die Werte auf der linken Seite höher ein.“*
>
> Zitiert aus [16]

„Individuen und Interaktionen mehr als Prozesse und Werkzeuge“ [16]: Dabei betont die agile Bewegung, dass Personen und Interaktionen eine höhere Bewertung, als Prozesse und Tools, bekommen. Wodurch Teamarbeit und Kommunikation durch ständige Überprüfungen/ Anpassungen verbessert werden. Um Vertrauen, Respekt und Transparenz im Team herstellen zu können, müssen Führungskräfte eine positive Austragung von Konflikten ermöglichen [1].

„Funktionierende Software mehr als umfassende Dokumentation“ [16]: Die Dokumentation in der Softwareentwicklung ist ein wichtiger Bestandteil, aber einer der Vorteile der agilen Softwareentwicklung, ist die Konzentration auf stets funktionierende Software. Dabei werden dem Kunden in regelmäßigen, kurzen Abständen kleine funktionsfähige Softwarepakete geliefert.

„Zusammenarbeit mit dem Kunden mehr als Vertragsverhandlung“ [16]: Mittels agiler Softwareentwicklung soll gewährleistet werden, dass der Kunde durch ständige Kommunikation mit der Geschäftsebene des Unternehmens am Ende eine Software bekommt, die er benötigt und nicht die er vor Anfang des Projektes in Auftrag gegeben hat. Dabei setzen agile Methoden, im Gegensatz zu klassischen Modellen, auf flexible Verträge. Dies setzt ein hohes Maß an Vertrauen, Flexibilität und gegenseitigem Respekt zwischen den Vertragsparteien voraus, welches vertraglich schwer regelbar ist [17].

„Reagieren auf Veränderung mehr als das Befolgen eines Plans“ [16]: Dadurch, dass sich während des Projektes Anforderungen verändern können, soll das agile Entwicklungsteam in der Lage sein, darauf schnellstmöglich zu reagieren. Dies ist durch kurze Entwicklungszyklen, wie sie in agilen Methoden verwendet werden, leichter möglich, als bei klassischen Entwicklungsmodellen, da es dabei vor den eigentlichen Änderungen zu einem wesentlich längerem Entscheidungsprozess (Change Management Prozess) kommt.

1.2.2 Prinzipien der agilen SW Entwicklung

Um die Befolgung der oben angeführten Werte hinter dem agilen Manifest sicherzustellen, wurden folgende zwölf Prinzipien verfasst: [16]

1. *„Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen“* [16]: Um herauszufinden, welche Software der Kunde wirklich benötigt, bekommt er in regelmäßigen Abschnitten funktionsfähige, kleinere Softwarepakete geliefert.

2. *„Heisse Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden“* [16]: Agile Softwareentwicklung sieht die kurzen Entwicklungszylen und somit die schnelle Anpassungsmöglichkeit bei Änderungswünschen als Wettbewerbsvorteil für Kunden. Bei großen Projekten und der daraus resultierenden langen Entwicklungsdauer, kann es z.B. durch veränderte Marktbedürfnisse oder Gesetzesänderungen zu notwendigen Anpassungen der Anforderungen während des Projektes kommen.

3. *„Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne“* [16]: Wie oben besprochen, sind kurze Entwicklungszyklen für beide Seiten enorm wichtig. Einerseits bekommt der Kunde einen Überblick, über schon funktionierende Software und kann Änderungswünsche öfters einbringen und andererseits hat das Entwicklungsteam den Vorteil, dass es sich durch regelmäßiges Feedback, hinfällige Arbeit erspart.

4. *„Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten“* [16]: Entwickler und Stakeholder (siehe Abschnitt 1.3.1.2) sollten täglich zusammenarbeiten und sich gegenseitig über Probleme und Erfahrungen austauschen, um daraus zu lernen. Dieses Prinzip wird in Scrum durch das *Daily Scrum Meeting* (Abschnitt 1.3.1.3) eingehalten.

5. *„Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen“* [16]: Agile Softwareentwicklung baut auf dem Prinzip, dass die Motivation eines Mitarbeiters im Unternehmen mittels Förderung und Vertrauen gesteigert werden kann. Aus diesem Grund können Entscheidungen von Personen getroffen werden, welche die Situation am besten kennen.

6. *„Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht“* [16]: Diese persönliche Kommunikationsart hat den Vorteil, dass dabei direkt auf Signale (Sprache, Mimik und Gestik) eingegangen werden kann, wodurch ein Informationsaustausch zwischen Teammitgliedern bzw. Management und Kunde ohne Missverständnisse stattfindet.

7. *„Funktionierende Software ist das wichtigste Fortschrittsmaß“* [16]: Eines der Hauptziele von agiler Softwareentwicklung ist zugleich auch ein Maß für den Projektfortschritt. Durch die regelmäßige Lieferung von lauffähigen Softwarepaketen kann der Kunde direkt den Fortschritt einschätzen. Bei klassischer Entwicklung werden meist unzählige KPIs (*Key Performance Indicator* - Leistungskennzahl) für die Messung des Projektfortschrittes verwendet, aber das wichtigste Kriterium für den Erfolg des Projektes außer Acht gelassen.

8. *„Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können“* [16]: Nachhaltige, gleichmäßige Entwicklung wird durch Vermeidung von Überarbeitung der Teammitglieder bewerkstelligt. Dadurch bleibt die Kreativität und Produktiviät über einen langen Zeitraum gewährleistet.

9. *„Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität“* [16]: Agile Prozesse legen Wert auf Mitarbeiter die eine uneingeschränkte Teamfähigkeit und fachliche Kompetenz besitzen, sowie ein Augenmerk auf gut strukturierte, wartbare und testbare Software legen.

10. *„Einfachheit – die Kunst, die Menge nicht getaner Arbeit zu maximieren – ist essenziell“* [16]: Ein weiteres Kriterium von Agilität ist die Konzentration auf die Entwicklung der derzeitigen Anforderungen des Kunden und die Außerachtlassung von möglichen Zukunftswünschen.

11. *„Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams“* [16]: Sich selbst organisierende Teams haben den Vorteil, dass die Motivation und Kreativität der Mitarbeiter ein höheres Level, als bei klassischen Prozessen, erreicht.

12. *„In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an“* [16]: Ein weiterer Aspekt agiler Methoden ist die stetige Weiterentwicklung von Teams, durch regelmäßige teaminterne Analysen von Problemen und Erfolgen vom jeweiligen Projektabschnitt. Dies wird am Beispiel von Scrum nach jedem Sprint mittels *Retrospektive Meetings* vollzogen. In herkömmlichen Softwareentwicklungsprozessen werden oftmals die Probleme und Herausforderungen nach einem Projekt, z.B. mittels *Lessons Learned Meeting* zusammengefasst, wodurch das aktuelle Projekt keinen Nutzen daraus ziehen kann.

1.2.3 Einsatzbereich von agiler Softwareentwicklung

Agile Softwareentwicklung ist eine interessante Alternative zu klassischen Entwicklungsprozessen. Dabei ist es jedoch so, dass der erfolgreiche Einsatz von agilen Methoden geprüft werden muss und vom Art des Projektes abhängt. Um diesem Thema genauer auf den Grund zu gehen, wurde im nächsten Abschnitt das „Stacey Complexity Model“ näher beschrieben (Abb. 1.2). Das „Stacey Complexity Model“ vergleicht folgende beiden Eigenschaften miteinander und unterteilt Projekte/Aufgaben in vier Kategorien (Simpel, Kompliziert, Komplex und Anarchie) [2]:

a) Unsicherheit der Requirements (Ordinate)

b) Unsicherheit der Technologie (Abszisse)

Simple Projekte/Aufgaben haben klar definierte Anforderungen mit einer bekannten Technologie, wie z.B. bei einer Serienproduktion.
Komplizierte Projekte sind Projekte bei denen das *Was?* und *Wie?* nicht mehr eindeutig geregelt ist.
Projekte aus der Kategorie *Anarchie* kann man als chaotisch bezeichnen. Das heißt, dass weder die Anforderungen, noch die Technologie bekannt ist und somit die Abwicklung solcher Aufgaben oder Projekte als äußerst schwierig betitelt werden kann. Beispiel: Autonomes Fahren in den Anfangsjahren
Komplex: Im letzten und zugleich größten Bereich des Modells, lässt sich agile Software-

entwicklung am Besten bzw. Sinnvollsten einsetzen. Einerseits sind die Requirements am Beginn des Projektes nicht vollständig festgelegt, wodurch es immer wieder zu Anforderungsänderungen kommen kann und andererseits kann sich die Art der Umsetzung im Laufe des Projektes noch entwickeln. Beispiel: Entwicklung von Software bei der Einführung von neuen Abgasnormen.

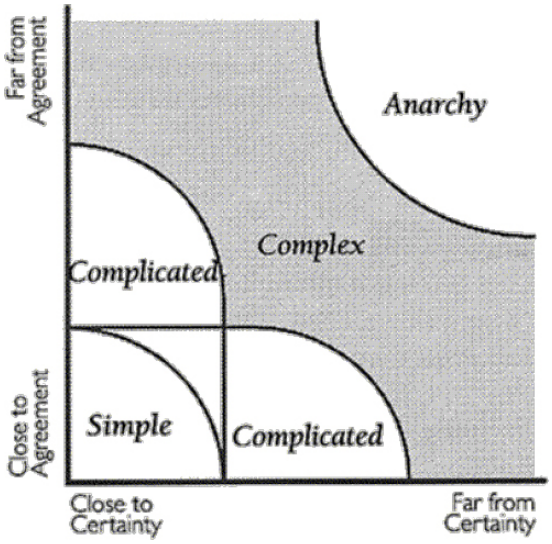


Abb. 1.2: Stacey complexity model: Unterteilung von Projekten und Aufgaben in vier Kategorien (Simpel, Kompliziert, Komplex und Anarchie). Ordinate: Unsicherheit der Anforderungen, Abszisse: Unsicherheit der Technologie [2]

1.3 Agile Methoden

Im Laufe der Zeit haben sich eine Vielzahl von agilen Methoden entwickelt (Abb. 1.1). Für diese Arbeit wurden zwei agile Methoden (Scrum, Kanban) ausgewählt und näher analysiert, da sie sich in den letzten Jahren als die am häufigst verwendeten Methoden herauskristallisiert haben.
Laut einer Umfrage von Kugler Maag Cie GmbH aus dem Jahr 2015 verwenden 79% aller Unternehmen aus der Automobiltechnikbranche, die agile Entwicklungsmethoden anwenden, Scrum und 55% Kanban, wobei sich Kanban oft als Alternative oder Ergänzung zu Scrum angeboten und im Vergleich zum Jahr davor deutlich an Boden gut gemacht hat (Abb. 1.3) [3]. Da *Continuous Integration* keine agile Methode, sondern eine einzelne agile Praktik darstellt, bestätigt diese Studie die Annahme, dass Scrum und Kanban die derzeit meist verwendetesten agilen Methoden sind.

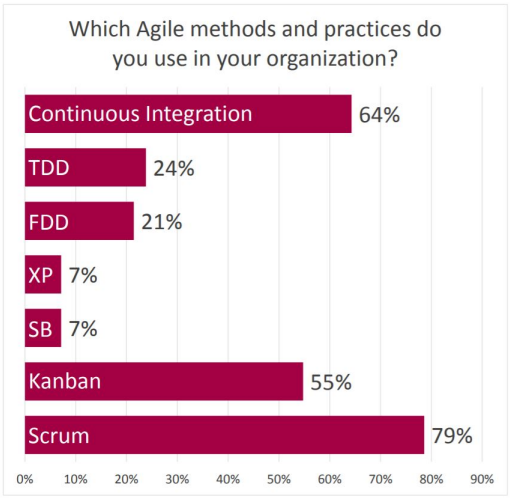


Abb. 1.3: Kugler Maag CIE - Umfrage: Agile Methoden und Praktiken im Anwendungsvergleich. Die Angabe der Häufigkeit, in der die Methoden/Praktiken in den befragten Organisationen der Automobiltechnik angewendet werden, wurde in Prozent angegeben. TDD (Test Driven Development), FDD (Feature Driven Development), XP (Extreme Programming), SB (Scrumban) [3]

Als zusätzliche Studie wurde eine Umfrage aus der 12.Ausgabe des „State of Agile Report“ von VersionOne verwendet, die besagt, dass 58% der befragten agilen Unternehmen aus unterschiedlichen Branchen Scrum und 5% Kanban verwenden (Abb.1.4) [4]. Aufgrund dessen und der Tatsache, dass die dazwischen liegenden Methoden, Kombinationen aus agilen Methoden sind, wird die getroffene Auswahl von Scrum und Kanban bekräftigt.

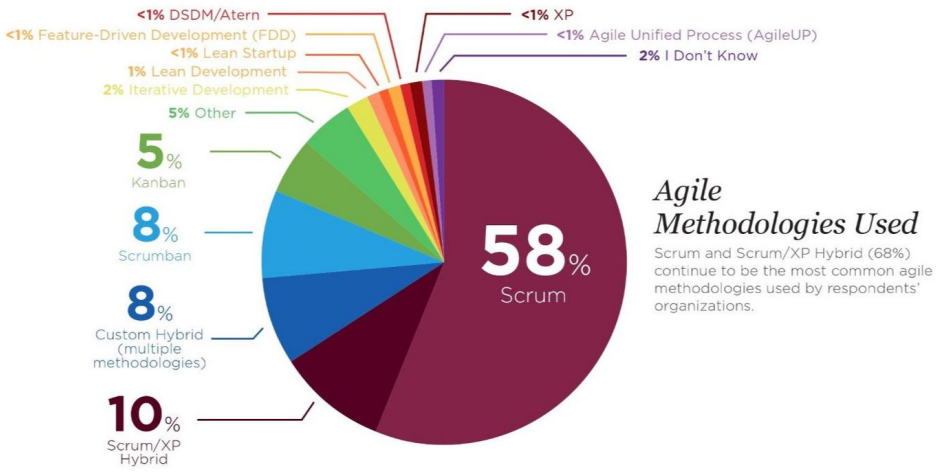


Abb. 1.4: VersionOne Inc. - Umfrage: Agile Methoden im Anwendungsvergleich. Die Angabe der Häufigkeit, in der die Methoden in den befragten Organisationen aus unterschiedlichen Branchen angewendet werden, wurde in Prozent angegeben. XP (Extreme Programming), DSDM (Dynamic Systems Development Method) [4]

1.3.1 Scrum

Das Wort Scrum ist ein Begriff aus dem Englischen und bedeutet Menschenauflauf oder Gedränge. Davon abgeleitet wurde der Scrum-Spielzug aus dem Rugby-Sport, in dem sich Spieler beider Mannschaften nah aneinandergedrängt auf dem Rasen befinden. Die Spuren der entgültigen Namensgebung der agilen Methode Scrum von Jeff Sutherland und Ken Schwaber sind darauf zurückzuführen. [18]
Scrum wird nicht als klassischer Prozess, sondern eher als Rahmenwerk angesehen, in dem Prozesse und Techniken eingesetzt werden. Es besteht aus kleinen Scrum Teams mit den zugehörigen Rollen, Ereignissen und Artefakten. Wie man in den nächsten Abschnitten erkennen kann, wird auf die folgenden drei Kriterien besonders Wert gelegt:

- Keine Veränderungen/Anpassungen während eines Sprints!
- Am Ende eines Sprints steht ein lieferbares Teilprodukt!
- Häufiger und wiederholter Erfolg schafft Vertrauen und Motivation!

1.3.1.1 Lebenszyklus

In folgender Abbildung (Abb. 1.5) ist der generelle Lebenszyklus von Scrum ersichtlich, wobei die einzelnen Rollen, Ereignisse und Artefakte im nächsten Abschnitt genauer erläutert werden. Grundsätzlich werden beim Start des Projektes alle erwünschten Produkteigenschaften im sogenannten *Product Backlog* zusammengefasst, woraus im nächsten Schritt ein *Sprint Backlog* mit mehreren *Backlog tasks*, vom Entwicklungsteam ausgewählt wird. Dieses *Sprint Backlog* soll in einem Sprint, welcher im Schnitt ein bis vier Wochen dauern soll, abgearbeitet werden, wobei es tägliche *Scrum Meetings* geben muss. Während eines Sprints ist das Entwicklungsteam von äußeren Einflüssen oder Veränderungen geschützt und kann somit die einzelnen Tasks in Ruhe abarbeiten. Am Ende eines Sprints werden mittels *Sprint Review* die enthaltenen Eigenschaften des lieferbaren Produktes präsentiert und über Erfahrungen bzw. Probleme diskutiert (*Sprint Retrospective Meeting*). [5]

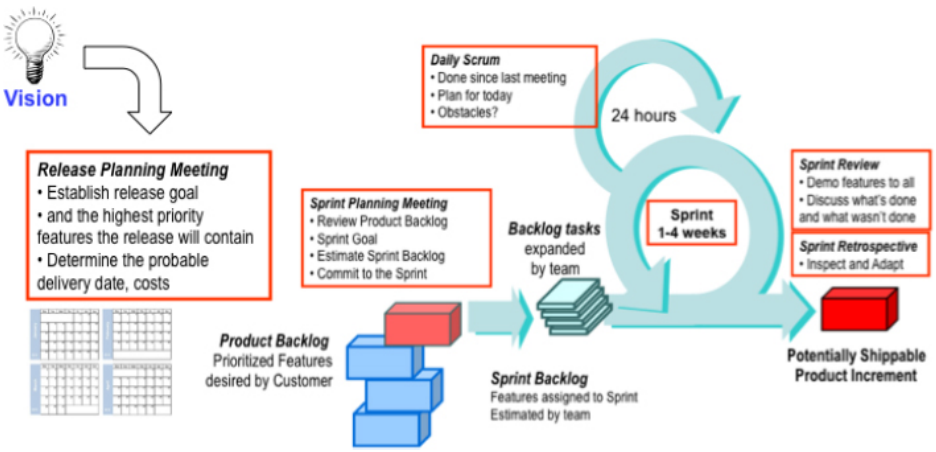


Abb. 1.5: Lebenszyklus des agilen Rahmenwerks Scrum mit aus Handlungen entstandenen Ergebnissen oder Methoden, den sogenannten Artefakten (z.B. Product Backlog) und Ereignissen (rot umrandet, z.B. Sprint Planning Meeting) [5]

1.3.1.2 Rollen

Scrum besteht mit dem *Product Owner*, dem Entwicklungsteam und dem *Scrum Master*, aus drei unterschiedlichen Rollen [5]. Zusätzlich zu diesen Rollen gibt es weitere optionale Rollen, wie unter anderem der *Quality Product Owner*, *Safety Product Owner* und *Agile Coach*, welche für sicherheitsrelevante Branchen von großer Notwendigkeit sind [19].

Product Owner: Der *Product Owner* ist für die Übermittlung der Vision des Kunden an das Entwicklungsteam verantwortlich. Er definiert und priorisiert die *Backlog tasks* des *Product Backlog*s, und somit die Funktionen des Produktes, und kann über das Release-Datum bzw. den Inhalt des Releases bestimmen. Des Weiteren ist er in der Lage, Veränderungen von Funktionen des Produktes für den nächsten Sprint einzubringen. [20]

Entwicklungsteam: Das Entwicklungsteam, welches aus 5-10 Personen besteht, die sich und ihre Arbeit selbst organisieren, sollte funktionsübergreifend sein, d.h. die Teammitglieder sollten in ein bis zwei Gebieten ihre Spezialitäten und in allen Bereichen Grundkenntnisse besitzen [20]. Sie wählen selbstständig das *Sprint Backlog* aus, entwickeln das Produkt und präsentieren am Ende des Sprints die Resultate dem *Product Owner* im *Sprint Review* [5].

Scrum Master: Der Scrum Master ist verantwortlich für die Führung, Verwaltung und das Coaching des Teams und des *Product Owner*s hinsichtlich Scrum. Er schützt das Team vor externen Störungen und beseitigt tägliche Hindernisse. Weiters ist er für die Umsetzung und Einhaltung von Scrum Ereignissen, wie *Daily Scrum-* oder *Sprint Planning Meetings*,

zuständig. [20]

Quality Product Owner: Da bei einer hohen Anzahl von Projekten oder einzelnen Großprojekten die Qualitätssicherung nicht auf der Strecke bleiben soll, fordern diverse Standards wie Automotive SPICE, aus der Automobiltechnik, eine unabhängige Qualitätssicherung. Bei der Verwendung von Scrum hat sich daher die Einführung einer zusätzlichen Rolle, nämlich die des *Quality Product Owner*s durchgesetzt, der für die Umsetzung der Qualitätsstrategie und der Qualitätsziele des Unternehmens zuständig ist. Damit wird sichergestellt, dass sich Qualitsaspekte, wie z.B. Wartbarkeit oder Usability, im *Product Backlog* befinden. [21]

Safety Product Owner: Ähnlich wie beim *Quality Product Owner*, ist der *Safety Product Owner* für die Sicherstellung von Sicherheitsaspekten im *Product Backlog*, welche in Standards wie ISO 26262 - Funktionale Sicherheit für Straßenfahrzeuge - gefordert werden, zuständig. Die Zusammenarbeit zwischen einem Manager für funktionale Sicherheit („Functional Safety Manager“) und den agilen Teams wird dadurch gewährleistet. [22]

Agile Coach: Wie in [23] beschrieben, besitzt die Rolle des *Agile Coach*es mehrere Aufgaben. Im Wesentlichen ist er für die Beobachtung, Ausbildung, Moderation und Unterstützung des Teams zuständig. Des Weiteren gibt er dem Team aus den gewonnenen Wahrnehmungen visuelles und verbales Feedback.

1.3.1.3 Ereignisse

Wie im Lebenszyklus von Scrum (Abb. 1.5) ersichtlich werden für die geregelte Durchführung mehrere Ereignisse, welche jeweils eine zeitlich vorgeschriebene Dauer (*Time Box*) besitzen, verwendet. Das Zentrum von Scrum ist der Sprint, eine *Time Box*, die während ihrer Laufzeit weder verkürzt noch verlängert werden darf. Rund um diesen Sprint werden einige vorausgesetzte Meetings (*Sprint Planning*, *Daily Scrum Meeting*, *Sprint Review* und *Sprint Retrospektive*) und das optionale *Release Planning Meeting* durchgeführt [5]. Zusätzlich dazu wird während des Sprints eine *Product Backlog*- Verfeinerung durchgeführt, die nicht mehr als 10% der Kapazität des Teams in Anspruch nehmen soll [24].

Release Planning Meeting: Im *Release Planning Meeting* werden die Ziele und Funktionen mit höchster Priorität für das folgende Release vom *Product Owner* präsentiert. Für die Planung der folgenden Releases werden den einzelnen Funktionen (*User stories*) Punkte

(*Story Points*) zugeordnet und somit dessen Aufwand geschätzt. Je nach Geschwindigkeit des Teams (*Velocity*) kann damit die Anzahl der *User stories* eines Sprints und somit die Anzahl der Sprints eines Releases bestimmt werden. [5]

Sprint: Innerhalb eines Sprints, welcher maximal 4 Wochen dauern darf, soll jeweils ein potentiell auslieferbares Produkt-Inkrement hergestellt werden. Der Sprint besteht aus einem *Sprint Planning Meeting*, Entwicklungszeit, *Daily Scrum Meetings*, dem Sprint Review und der Sprint Retrospektive. [24]

Im Sprint Planning Meeting werden vom gesamten Scrum Team zwei Fragestellungen beantwortet: Was ist das Ziel des Sprints? Wie können wir dieses Sprint Ziel erreichen? Dabei beschreibt der *Product Owner* das Ziel und die priorisierten *Product Backlog*-Einträge, die bei Abschluss das Ziel erreichen würden. Daraus wählt das Entwicklungsteam, eine machbare Anzahl von Einträgen für den nächsten Sprint und erarbeitet gemeinsam mit dem Scrum Team das Sprintziel. Der nächste Schritt befasst sich mit der Entscheidung des Entwicklungsteams, wie sie das vereinbarte Produkt-Inkrement (Sprint Ziel) erreichen werden. Dafür wird das *Sprint Backlog* erstellt, welches aus den geplanten *Product Backlog* Einträgen und einer Liste von Tasks besteht, die für dessen Umsetzung notwendig sind. [24]

Das Daily Scrum Meeting wird täglich, während eines Sprints zur selben Uhrzeit und am selben Ort durchgeführt. Dabei werden folgende drei Fragestellungen von den Teammitgliedern beantwortet:

- Was habe ich gestern für das Erreichen des Sprint Ziels geleistet?
- Was werde ich heute erledigen?
- Welche Hinternisse oder Risken werden mich davon abhalten?

Aus diesem Grund wird die Kommunikation zwischen Teammitgliedern verbessert, Wissensstand ausgetauscht und zu beseitigende Hinternisse identifiziert. [24]

Als Backlog Refinement wird der Vorgang der Verfeinerung des *Product Backlogs* bezeichnet, indem Einträge verändert, Schätzungen erstellt oder die Priorisierung der Einträge verändert werden kann. Dieser Prozess sollte nicht mehr als 10% der Sprintkapazität in Anspruch nehmen. Der *Product Owner* kann bei Verständnisfragen zu den einzelnen

Einträgen zur Seite stehen, aber die Schätzungen werden von Entwicklungsteam getroffen. Durch die Verfeinerung soll sichergestellt werden, das die wählbaren Einträge den Zustand „ready“ besitzen und somit im folgenden Sprint die Kriterien des *Definition of Done (DoD)* erfüllt werden können. [24]

Das Sprint Review Meeting wird am Ende des Sprints zwischen dem Scrum Team und wichtigen Stakeholdern abgehalten. Dabei präsentiert das Entwicklungsteam die Ergebnisse des Sprints und der Product Owner den aktuellen Stand des *Product Backlogs*. Weiters werden in diesem Meeting die nächsten Schritte bzw. etwaige Änderungen des *Product Backlogs* besprochen und dadurch ein Input für das nächste *Sprint Planning Meeting* geliefert. [24]

Beim Sprint Retrospektive Meeting wird der letzte Sprint rückblickend analysiert, um herauszufinden, was gut verlaufen ist, wobei es Probleme gab und welche Punkte im nächsten Sprint verändert werden sollten. Dabei wird ein Aktionsplan erarbeitet, der im folgenden Sprint umgesetzt und in der nächsten Sprint Retrospektive am Erfolg überprüft wird. [5]

1.3.1.4 Artefakte

Wie im Scrum Guide [24] beschrieben repräsentieren die Artefakte „*[...] die Arbeit oder Wert, um Transparenz sowie Möglichkeiten zur Überprüfung und Anpassung zu schaffen*“. Im Allgemeinen spricht man bei einem Artefakt von einem Ergebnis, das durch eine vorherige Handlung entstanden ist, wie z.B. einer Liste oder ein Grafik, oder von der Handlung selbst. Dabei gibt es zusätzlich zu den drei in Scrum beschriebenen Artefakten (*Product Backlog*, *Sprint Backlog* und Produkt Inkrement), noch weitere in der Praxis verwendete Artefakte, welche auf den nächsten Seite kurz beschrieben werden.

Product Backlog: Das *Product Backlog* ist eine Liste von Anforderungen für die Entwicklung eines Produktes, die vom *Product Owner* priorisiert wird [20]. Es ist zu Beginn des Projektes nicht im Entferntesten vollständig und es entwickelt sich stetig weiter, wodurch es auch als dynamisches Artefakt bezeichnet wird. Mittels *Backlog Refinement* können dessen Inhalte kontinuierlich verändert, neu priorisiert oder entfernt werden. [24]

Sprint Backlog: Wie in Abschnitt 1.3.1.3 schon kurz erwähnt wurde, besteht das Sprint Backlog aus allen für den Sprint ausgewählten Backlog Items und einem aus Tasks beste-

henden Plan für die erfolgreiche Umsetzung des Sprintziels und des Produkt Inkrements. Es wird vom Entwicklungsteam in Scrum meist als Taskboard geführt. [25]

Produkt Inkrement: Das Produkt Inkrement besteht aus den funktionsfähigen Implementierungen (DoD - erfüllt) des aktuellen Sprints und allen Inkrementen aus den vorherigen Sprints. Dabei müssen alle Funktionen miteinander funktionieren und ergeben somit nach jedem Sprint ein potentiell auslieferbares Produkt. [25]

User Story: Eine *User Story* ist eine Art von Beschreibung einer Produkteigenschaft, welche in Scrum ein *Product Backlog* Item darstellt. Diese *User Story* mit spezieller Formatierung (Rolle, Beschreibung und Nutzen in einem Satz enthalten z.B. „Als Bibliothekar möchte ich auf der Website Bücher nach Publikationsjahr suchen können“) wird in Kombination mit einer genaueren Beschreibung und der Definition der Akzeptanzkriterien auf sogenannten *Story Cards* geschrieben. Der Aufwand der User Stories wird mit *Story Points* geschätzt [25].

Impediment Backlog: Im *Impediment Backlog* werden alle Hindernisse (*Impediments*), die dem Team das Erreichen des Sprintziels erschwert oder davon abgehalten haben, vom *Scrum Master* aufgelistet und zu lösen versucht, um dem Entwicklungsteam die Arbeit in Zukunft zu erleichtern [25]. Logistische oder infrastrukturelle Probleme könnten als Beispiel genannt werden.

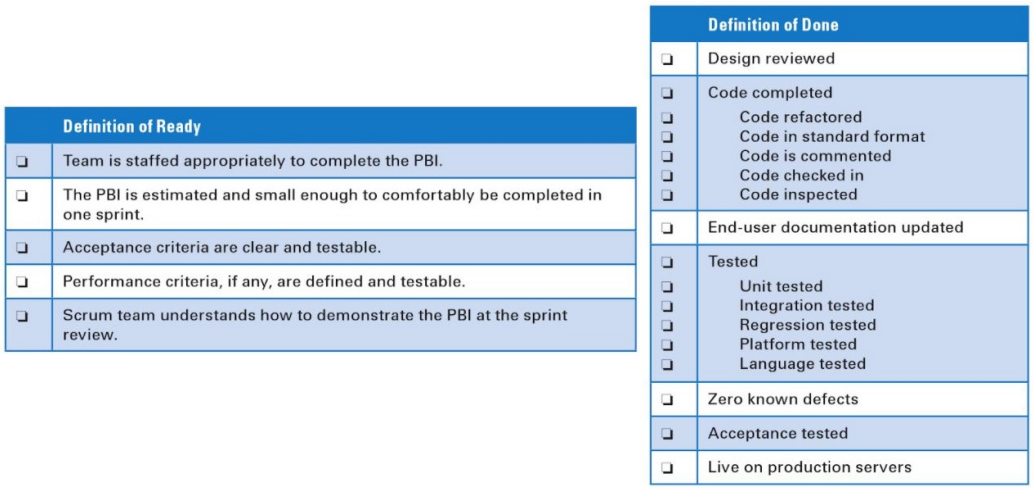
Taskboard: Ein Taskboard wird vom Entwicklungsteam dazu verwendet, den Fortschritt der Aufgaben für jede Funktion des *Sprint Backlogs* zu verfolgen. Dabei werden die einzelnen Aufgaben im *Daily Scrum Meeting* zwischen den Spalten (Minimum: „offen“, „in Arbeit“, „Fertig“) des Taskboards verschoben. [5]

Sprint Burndown Chart: Der *Sprint Burndown Chart* ist eine grafische Darstellung der noch zu erledigenden Arbeit des laufenden Sprints. Dabei wird der verbleibende Aufwand des Sprints (Tasks, *Story Points*), in Relation zur verbleibenden Zeit bis zum Ende des Sprints gestellt. [5] Dieser Chart wird nach jedem *Daily Scrum Meeting* vom Team auf den aktuellen Stand gebracht [25].

Velocity Chart: Beim *Velocity Chart* wird die Anzahl der erledigten Story Points der Sprintnummer gegenübergestellt, um einen Überblick über die Geschwindigkeitsentwicklung des Teams zu bekommen. Dies erleichert die Planung der nächsten Sprints und gibt dem Product Owner einen Ausblick über einen möglichen nächsten Liefertermin. [25]

Sprint-Ziel: Das Sprint-Ziel wird im *Sprint Planning Meeting* vom Entwicklungsteam gemeinsam mit dem *Product Owner* beschrieben [24]. Um zu verdeutlichen wie Sprintziele formuliert werden können, hierzu ein Beispiel: „Implementierung der Login Funktion im aktuellen GUI (Graphical User Interface).“

Definition of Ready/Done: *Definition of Ready (DoR)* beschreibt den Zustand eines Backlog Items, bei dem das Entwicklungsteam ihre Arbeit beginnen könnte. Nach der Durchführung des *Backlog Refinements* sollten die obersten Einträge des *Product Backlogs* auf „ready“ gesetzt sein, womit sie im nächsten *Sprint Planning Meeting* ausgewählt und abgearbeitet werden können. Der Zustand „done“ kennzeichnet jene Backlog Items die am Ende eines Sprints als erfolgreich abgeschlossen gelten. Erfüllen alle Items des *Sprint Backlogs* die Kriterien von *DoD*, kann das Produkt Inkrement als potentiell lieferbar deklariert werden. *DoD* und *DoR* sind Checklisten von Aktivitäten, die vor dem Setzen des Zustands abgeschlossen sein müssen (siehe Abb. 1.6). [6]
Die beiden Definitionen werden zu Beginn eines Projektes grundsätzlich festgeschrieben und können während des Projektes angepasst werden. Zu Beachten gilt jedoch, dass die Kriterien von *DoD* während eines Sprints nicht verändert werden dürfen. [19]



	Definition of Ready
❑	Team is staffed appropriately to complete the PBI.
❑	The PBI is estimated and small enough to comfortably be completed in one sprint.
❑	Acceptance criteria are clear and testable.
❑	Performance criteria, if any, are defined and testable.
❑	Scrum team understands how to demonstrate the PBI at the sprint review.

	Definition of Done
❑	Design reviewed
❑	Code completed
❑	Code refactored
❑	Code in standard format
❑	Code is commented
❑	Code checked in
❑	Code inspected
❑	End-user documentation updated
❑	Tested
❑	Unit tested
❑	Integration tested
❑	Regression tested
❑	Platform tested
❑	Language tested
❑	Zero known defects
❑	Acceptance tested
❑	Live on production servers

Abb. 1.6: Beispiele der beiden agile Artefakte *Definition of Done (DoD)* und *Definition of Ready (DoR)* in Form von Checklisten [6]

Planning Poker: *Planning Poker* mit Karten ist eine Methode zur Schätzung des Aufwands von *User stories* des *Product Backlogs*. Diese Methode basiert auf der Breitband Delphi Methode, bei der sich mehrere Experten für die Schätzung der Aufgaben abstimmen. Beim *Planning Poker* werden Spielkarten verwendet, dessen Werte (*Story Points*) exponentiell ansteigen (z.B. 1, 2, 4, 8, 16). Am Beginn wird über die einzelnen User Stories solange

diskutiert, bis alle Teammitglieder ein gemeinsames Verständnis darüber besitzen und im Anschluss daran jedes Mitglied eine Schätzung verdeckt durchführen kann. Nach dem Aufdecken der verdeckten Spielkarten, werden die höchsten und niedrigsten Schätzungen, bis zur Einigung auf eine gemeinsame Wertung, besprochen. [19]

1.3.2 Kanban

Ein Kanban System wird laut David J. Anderson grundsätzlich dazu verwendet, die Menge an begonnener Arbeit auf eine feste Anzahl zu beschänken, um dadurch das Team nicht zu überlasten und den Teammitgliedern die Möglichkeit zu geben ihre Aufgaben erfolgreich abzuschließen. Die Hauptkomponenten von diesem Ansatz, der zur kontinuierlichen Verbesserung führen soll, beinhalten drei Prinzipien und fünf Praktiken. [26]

1.3.2.1 Die Prinzipien von Kanban:

1. *Starte mit dem woran du gerade arbeitest:* Das Kanban System schreibt keine spezielle Prozedur vor, es wird jedoch vorausgesetzt, dass ein bestehender Prozess vorhanden ist, auf dem das System aufgesetzt werden kann [26].

2. *Respektiere alle aktuellen Prozesse, Rollen und Verantwortlichkeiten:* Kanban sieht, dass vorhandene Prozesse, Rollen und Verantwortlichkeiten zum Erfolg des Unternehmens beigetragen haben können. Aus diesem Grund ist es in Kanban nicht zwingend notwendig in diesem Bereich Änderungen durchzuführen, es wird aber auch nicht verboten. Im Falle von Veränderungen bevorzugt Kanban inkrementelle Veränderungsschritte. [27]

3. *Verfolge inkrementelle und evolutionäre Verbesserung:* Der Kanban Ansatz ist ein Änderungsmanagement, der auf minimalen Widerstand aufgebaut ist. Dabei werden kontinuierliche inkrementelle, evolutionäre Anpassungen bevorzugt und auf weitreichende Veränderungen verzichtet. [27]

1.3.2.2 Hauptpraktiken von Kanban

1. Visualisierung des Arbeitsflusses: Dabei geht es darum, dass man den Arbeitsfluss verstehen muss, um erfolgreiche Verbesserungen einführen zu können. Durch diese Visualisierung der Arbeit werden Zusammenhänge sichtbar und bessere Entschei-

dungen getroffen. [27]
Die gebräuchliste Art der Visualisierung des Arbeitsflusses, ist die Verwendung von Kartenwänden (*Kanban Board*), Karten und Spalten. Die in kleinere Aufgaben unterteilten Funktionen, werden auf Karten niedergeschrieben und auf dem Kanban Board angebracht. Das *Kanban Board* besteht aus benannten Spalten, die den Status einer Aufgabe veranschaulichen sollen. (siehe Abb. 1.7)

2. Limitierung des Work in Progress (WIP): Die Limitierung der Anzahl an begonnenen Arbeiten entlastet das Team, senkt die Durchlaufzeiten und bewirkt einen kontinuierlichen Arbeitfluss. Dabei liegt der Fokus klar auf dem Abschluss von wenigen Arbeiten, als auf der gleichzeitigen Bearbeitung von Vielen. Um die Anzahl der parallel bearbeiteten Tasks zu begrenzen, bekommt jede Spalte des Kanban Boards ein individuelles WIP - Limit. [28]

3. Kontrolle des Arbeitsflusses: Da ein stetiger, schneller und vorhersehbarer Arbeitfluss bei Kanban eine große Rolle spielt, wird dieser durch mehrere Artefakte gesteuert und gemessen.
   Aufgrund dessen, dass nicht alle Aufgaben denselben Dringlichkeitsgrad haben, werden Aufgaben in sogenannte Serviceklassen oder Arbeitstypen eingeteilt. Diese bilden die Basis für *Service Level Agreements* (SLAs), wodurch den Kunden eine Sicherheit gegeben wird, dass Arbeiten einer Serviceklasse bzw. eines Arbeitstypen in einem definierten Zeitraum fertiggestellt werden. [28]
   Dies wird durch die Einführung von mehreren Zeilen (*Swim Lanes*) im Kanban Board implementiert [26]. Beispiele dafür können wie folgt lauten: „fast lane“, „Fixes Fälligkeitsdatum“, „restliche Aufgaben“.
   Ein weiterer wichtiger Punkt für die Steuerung ist die Festlegung von Akzeptanzkriterien für den Übergang in den nächsten Status. Dabei wird die Aufgabe nur als erledigt angesehen, wenn das Akzeptanzkriterium erfüllt wurde. [19]

4. Festlegung von Prozessregeln: Der Arbeitsfluss unterliegt in Kanban einer Menge von Regeln, die eingehalten oder gegebenfalls angepasst werden müssen, um den Fluss aufrecht zu erhalten [28]. Dabei werden unter anderem Richtlinien für personelle WIP Limitierung, die Verwendung des Kanbanboards oder die zuvor angesprochenen Akzeptanzkriterien, festgelegt [29].

5. Gemeinschaftliche Verbesserung des Flusses: Die fünfte Praktik umfasst die gemeinschaftliche Verbesserung des Arbeitflusses basierend auf ausgewerteten Metriken, den

Ergebnissen von täglichen Statusmeetings oder Retrospektivmeetings. Dabei können Verbesserungsmöglichkeiten unter anderem im gesetzten WIP- Limit auftreten. [29]

Die folgende Abbildung (Abb. 1.7) „One day in Kanban land“ beschreibt die Abarbeitung von Funktionen mittels Kanban Boards und limitierten WIPs. Dabei zeigt dieser Ablauf, wie die Limitierung von WIP, Probleme vom Team aufgedeckt und zu lösen versucht werden.

Im ersten Schritt (a) wird der Start eines Projektes und die Aufnahme der ersten beiden Tasks in den Status „Selected“(WIP = 2) dargestellt. Sobald ein Task für die Entwicklung bereit ist, kann das Entwicklungsteam die Aufgabe aufnehmen und diese im Falle des Abschlusses in den Bereich „Develop: Done“ verschieben (b). Aus diesem Grund kann Task A im nächsten Bereich „Deploy“ bearbeitet werden und das Entwicklungsteam eine weitere Aufgabe aus dem Produkt Backlog auswählen (c). Im Falle einer Fertigstellung eines weiteren Tasks im Entwicklungsbereich und dementsprechender Auswahl einer dritten Aufgabe, würde das WIP-Limit, aufgrund von Problemen im Auslieferungsteil des Kanbanboards, überschritten werden. Nun eilen die Teams, die aufgrund der WIP-Beschränkung nicht weiterarbeiten können, zu Hilfe und versuchen das Problem gemeinsam zu lösen, um den Arbeitsfluss wieder zu starten (d-f). Aufgrund der WIP-Limitierung wurde das Problem vom gesamten Team erkannt, gelöst und durch das Schreiben von Tests ein zukünftiges Auftreten verhindert. [7]

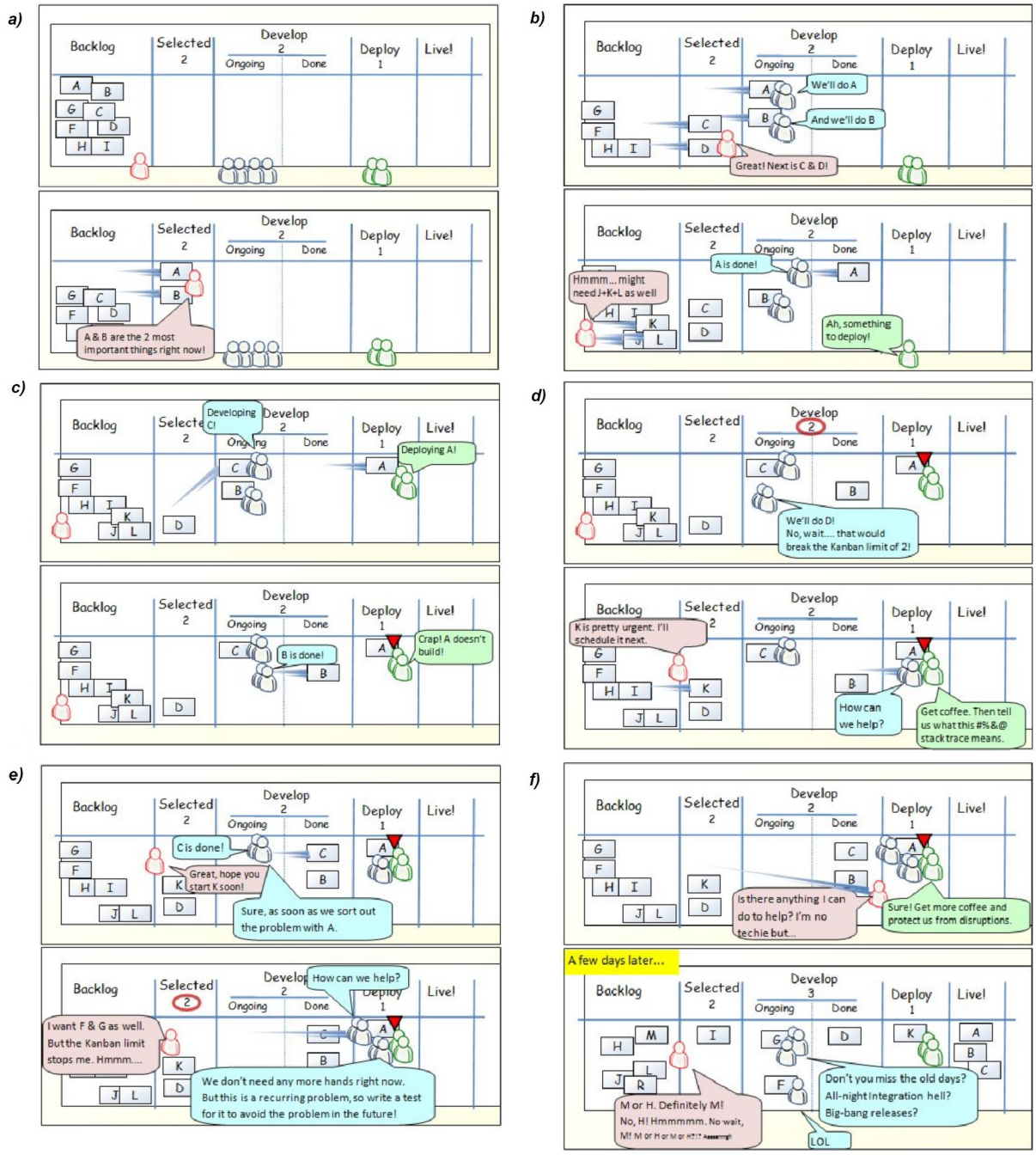


Abb. 1.7: „One day in Kanban land“[7]

1.4 Harmonisierte Normen

In diesem Abschnitt werden die beiden relevanten Standards für Softwareentwicklung im Automobil- und Medizintechnikbereich vorgestellt. Dabei handelt es sich einerseits im Automobiltechnikbereich um *Automotive SPICE - Process Assessment/Reference Model 3.1* ([8]), welches sich mit der Beurteilung und Verbesserung von bereits bestehenden Prozessen beschäftigt und Prozesse bzw. Prozessanforderungen im Prozessreferenzmodell (PRM), für die Entwicklung von Software in der Automobilbranche, festlegt. Automotive SPICE (ASPICE) ist eine etablierte branchenspezifische Variante des internationalen Standards ISO 15504 (SPICE) [30] und wird daher von vielen Automobiltechnikfirmen, u.a. auch in der AVL List GmbH verwendet.

Andererseits fiel die Wahl im Medizintechnikbereich auf den Standard *EN 62304:2006* ([31]), in dem ein Lebenszyklus für die Entwicklung und Wartung von Software in Medizinprodukten behandelt wird, der aus mehreren Prozessen, Aktivitäten und Aufgaben besteht.

1.4.1 Automotive SPICE - Process Assessment/Reference Model 3.1

Das Prozessassessmentmodell (PAM) aus *Automotive SPICE - Process Assessment/Reference Model 3.1* bewertet die Prozessfähigkeit mit einem zweidimensionalen Modell:

- Prozessdefinition: Geforderte Prozesse werden im PRM definiert und in Prozesskategorien eingeteilt.
- Reifegraddimension: Besteht aus Reifegradstufen, die in Prozessattribute unterteilt werden, die wiederum messbare Eigenschaften der Prozessfähigkeit liefern. Genaue Aufgliederung der Reifegradstufen (Stufe 1 bis 5) und ihre Attribute können in der Norm ([8]) entnommen werden.

Das PAM wählt Prozesse aus dem PRM und Ergänzungen mit Indikatoren aus. Diese Indikatoren ermöglichen dem Assessor eine Sammlung von Beweisen für die Bewertung von Prozessen gemäß den Reifegraddimensionen. Abbildung 1.8 zeigt den Zusammenhang zwischen PAM und PRM. [8] Da ein Vergleich zwischen den geforderten Prozessen und Aktivitäten von ASPICE und EN 62304 erstellt werden soll, wird in diesem Abschnitt nur auf die Struktur des PRM näher eingegangen.

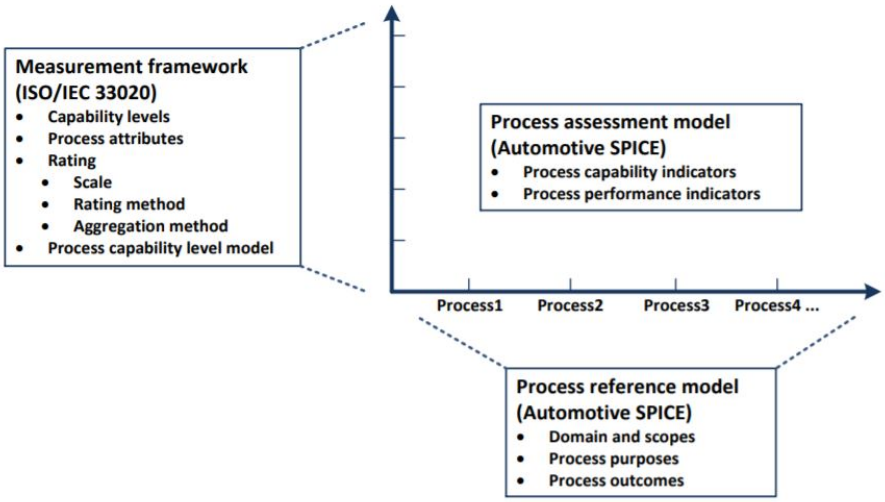


Abb. 1.8: Beziehung zwischen dem Prozessassessmentmodell (PAM), Prozessreferenzmodell (PRM) aus Automotive SPICE 3.1 [8] und dem Bewertungsrahmen aus ISO/IEC 33020:2015 [9].

1.4.1.1 Struktur des Prozessreferenzmodells (PRM)

Grundsätzlich wird das PRM von ASPICE 3.1 in folgende drei Prozesskategorien zusammengefasst, welche wiederum in Prozessgruppen, je nach Aktivität unterteilt werden (Abb. 1.9) [8].

- Primäre Prozesse im Lebenszyklus: Besteht aus Prozesse die einerseits vom Kunden beim Erwerb von Produkten des Lieferanten, und andererseits vom Lieferanten bei der darauffolgenden Reaktion und Produktlieferung an den Kunden verwendet werden können [8].

- Organisatorische Prozesse im Lebenszyklus: Prozesse die Prozess-, Produkt- und Ressourcen-Assets entwickeln, die bei der Verwendung in Projekten, der Organisation bei der Erreichung ihrer Unternehmensziele behilflich sind. Prozess-, Produkt- und Ressourcen-Assets sind Erfahrungen, Dokumente, Newsletter, Templates und Datenbanken, die Prozesse, Produkte und Ressourcen beschreiben und unterstützen [8].

- Unterstützende Prozesse im Lebenszyklus: Diese Prozesskategorie umfasst all jene Prozesse, die von allen anderen Prozessen im Lebenszyklus eingesetzt werden können [8].

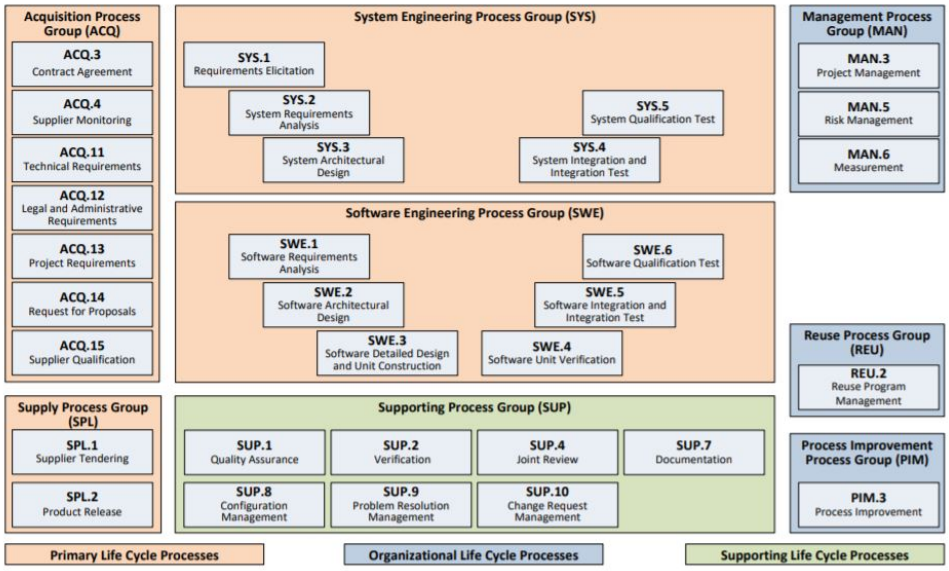


Abb. 1.9: Darstellung des Prozessreferenzmodells aus Automotive SPICE 3.1 mit den zugehörigen Prozesskategorien (Primäre Prozesse, Organisatorische Prozesse und unterstützende Prozesse), Prozessgruppen und Prozessen.[8]

Zusätzlich zum vollständigen Prozessreferenzmodell in Abbildung 1.9 gibt es eine beschränkte Auswahl an Prozessen aus ASPICE, die bei jedem Assessment durch Vertreter der Herstellerinitiative-Software (HIS) beurteilt werden. Dabei handelt es sich um den sogenannten „VDA Scope“, ehemals HIS-Scope, der als Minimalumfang an Prozessen für ein Assessment von der HIS ausgerufen und in Abbildung 1.10 angegeben wurde.

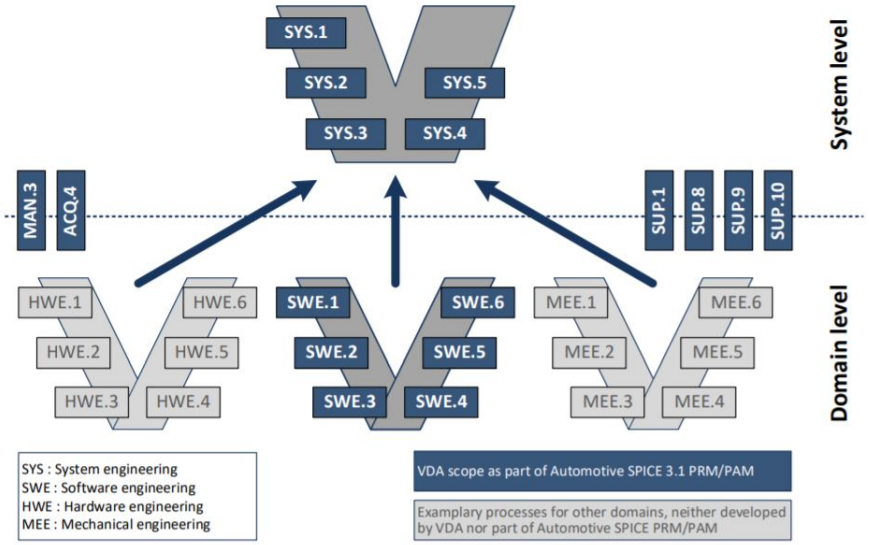


Abb. 1.10: Abbildung der Prozesse des VDA-Scopes (dunkelblau) der HIS aus *Automotive SPICE 3.1* [8]

Jeder Prozess ist in Automotive SPICE 3.1 in Tabellenform gegliedert und besteht aus den beiden Bereichen Prozessreferenzmodell (ID, Name, Ziel und Ergebnisse des Prozesses) und den Prozess-Performance-Indikatoren (*Base Practices*, *Output Work Products*) (Abb. 1.11). *Base Practices (BPs)* enthalten Aufgaben und Aktivitäten, die zur Erreichung des Prozessziels und zur Erfüllung der Prozessergebnisse erforderlich sind. [8]

Output Work Products sind Arbeitsprodukte dessen Verwendung zur Erreichung der *Base Practices* und des Prozessziels führen können. Dabei ist stets eine Verbindung zwischen *Output Work Products*, *Base Practices* und des Prozessziels in dieser Prozessbeschreibung enthalten.

Process reference model	**Process ID** **Process name** **Process purpose** **Process outcomes**	The individual processes are described in terms of process name, process purpose, and process outcomes to define the Automotive SPICE process reference model. Additionally a process identifier is provided.
Process performance indicators	**Base practices**	A set of base practices for the process providing a definition of the tasks and activities needed to accomplish the process purpose and fulfill the process outcomes
	Output work products	A number of output work products associated with each process *NOTE: Refer to Annex B for the characteristics associated with each work product.*

Abb. 1.11: Prozessbeschreibung aus Automotive SPICE 3.1 mit den beiden Bereichen Prozessreferenzmodell (ID, Name, Ziel und Ergebnisse des Prozessen) und Prozess-Performance-Indikatoren (*Base Practices* und *Output Work Products*).[8]

1.4.2 Medizingeräte-Software - Software-Lebenszyklus-Prozesse (EN 62304: 2006)

Der Standard *EN 62304:2006* legt einen Rahmen von Lebenszyklus-Prozessen mit Aktivitäten und Aufgaben für die sichere Entwicklung und Wartung von Software für Medizinprodukte fest, wobei Anforderungen für diesen Lebenszyklus definiert werden. Dabei gilt diese Norm bei Software die ein eigenständiges Medizinprodukt oder ein integrierter Bestandteil des fertigen Medizinproduktes ist. [31]

Da bei dieser Masterarbeit die Wartung von Software eine untergeordneter Rolle spielt, wird hierbei auch nur der Software-Entwicklungs-Lebenszyklus näher betrachtet.

1.4.2.1 Lebenszyklus der Softwareentwicklung von Medizinprodukten

In Abbildung 1.12 wird ein Überblick über den Softwareentwicklungs-Lebenszyklus mit den Prozessen und Aktivitäten unter EN 62304 dargestellt.

Dieser Lebenszyklus besteht aus dem Software-Entwicklungs-Prozess mit mehreren Aktivitäten (§ 5.x), dem Software-Riskomanagement-, Software-Konfigurationsmanagement- und dem Problemlösungs-Prozess für Software. [31] Auf einzelne ausgewählte Aktivitäten wird in weiterer Folge näher eingegangen.

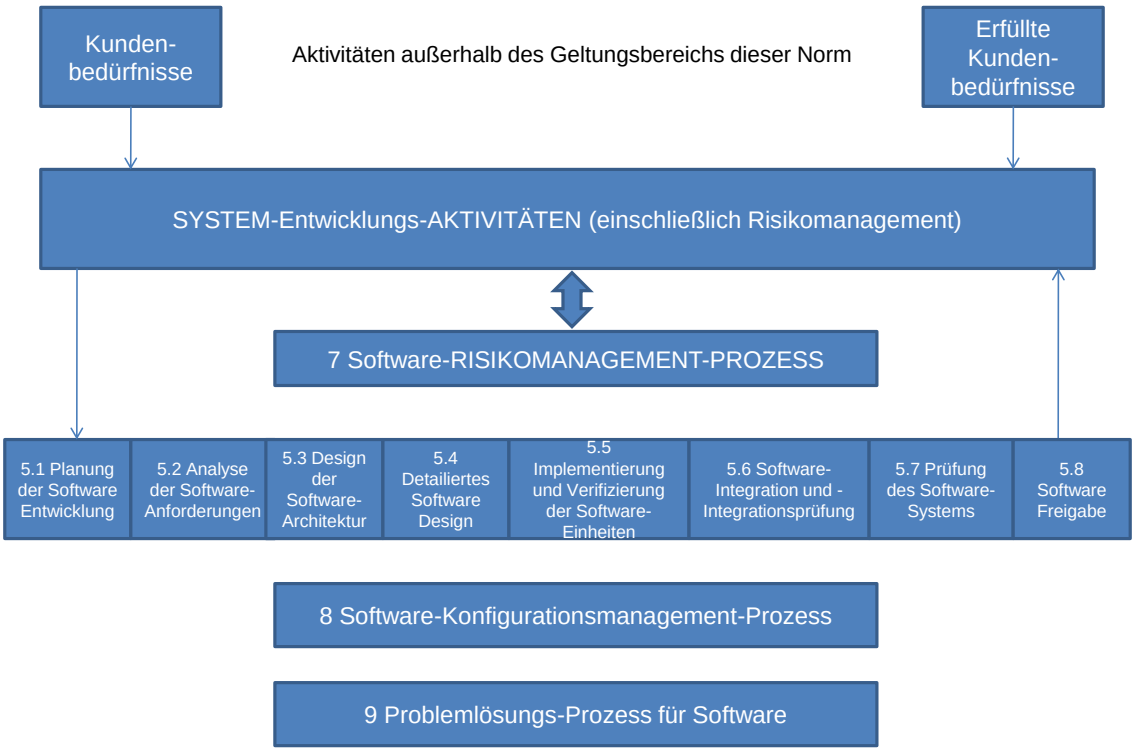


Abb. 1.12: Überblick über die Softwareentwicklungsprozesse und Aktivitäten aus EN 62304:2006

In Abbildung 1.13 sind die Aktivitäten § 5.1 - § 5.8 des Softwareentwicklungsprozesses für Medizinprodukte aus EN 62304 als V-Modell ersichtlich. Dabei muss erwähnt werden, dass dies nur den untersten Bereich des V-Modells aus dem Lebenszyklus der Softwareentwicklung von Medizinprodukten darstellt und somit Systementwicklungsaktivitäten, der Softwarerisikomanagementprozess (§ 7), der Softwarekonfigurationsmanagementprozess (§ 8) und der Problemlösungsprozess für SW (§ 9) darin nicht enthalten sind.

Grundsätzlich ist das V-Modell in zwei Bereiche aufgeteilt, den Spezifizierungs- und Entwicklungsphasen auf der linken Seiten und den Testphasen auf der rechten Seite. Jeder Spezifizierungs- oder Entwicklungsphase ist eine Testphase gegenübergestellt, wie z.B. an der Spitze des V's die eigentliche Entwicklung der Software die Verifizierung dieser Software gegenübergestellt bekommt. Die schwarzen Pfeile zeigen die möglichen Richtungen zwischen zwei Phasen an. Der Pfeil entgegen der normalen Richtung des V-Modells (z.B. von § 5.3 auf § 5.2) kennzeichnet die Kontrolle der Einhaltung der vorherigen Phase (Verifikation). Der Schritt zwischen zwei gegenüberliegenden Phasen, stellt einerseits die Fehlersuche (⟵) und andererseits die Generierung von Testfällen (⟶) dar (Validierung).

Einer der größten Vorteile des V-Modells in der Softwareentwicklung sind hohe Qualitätsstandards, da jede Anforderungsebene einzeln geprüft und unvollständige Spezifikationen frühzeitig erkannt und behoben werden können. Dem gegenüber steht die relative lange Entwicklungsdauer und geringe Flexibilität des Modells. [32]:

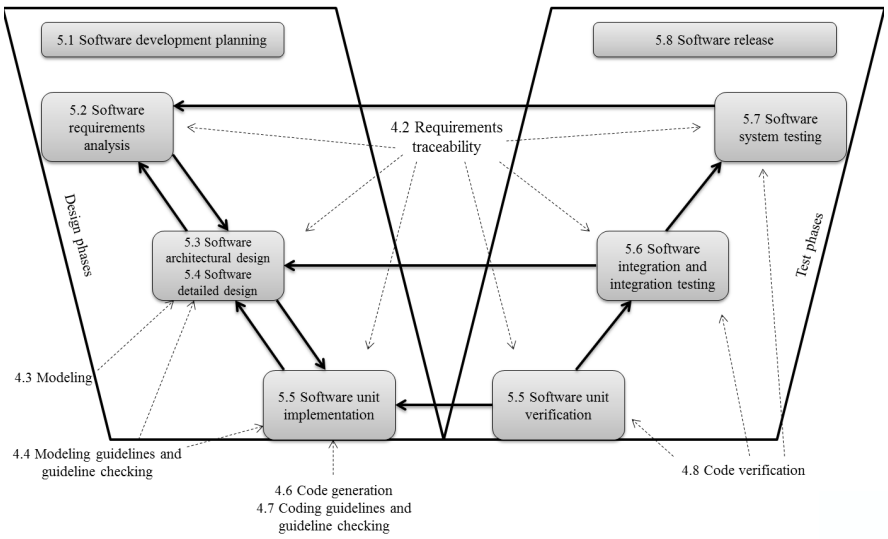


Abb. 1.13: V-Modell für die Entwicklung von Medizingerätesoftware laut EN 62304 [10]. Die Methoden 4.2 - 4.8 stammen aus der Quelle der Abbildung und sind für diese Arbeit nicht relevant.

2 Aufgabenstellung

In dieser Masterarbeit wird analysiert, wie agile Softwareentwicklung in sicherheitskritischen Industriebereichen funktionieren könnte und es werden die daraus resultierenden Problemfelder aufgezeigt. Um einen besseren Überblick über das Verhalten dieser Art der Softwareentwicklung in Industriebereichen mit hohen Sicherheitsanforderungen zu bekommen, werden zwei verschiedene Branchen - Automobilindustrie und Medizintechnik - eingehend analysiert. Dabei liegt das Ziel dieser Untersuchung darin, dass Probleme beider Branchen identifiziert und auf Ähnlichkeiten analysiert werden.

Für den zweiten Teil der Arbeit werden Lösungsvorschläge zur Bewältigung der Probleme entwickelt.

Um das oben angegebene Ziel der Arbeit zu erreichen, wurden folgende drei Forschungsfragen definiert:

1. Welche Probleme ergeben sich bei der Anwendung von agilen Software- Entwicklungsmethoden für die Entwicklung von Software in der Automobil- und Medizintechnik?

2. Welche gemeinsamen Problemfelder treten bei der Nutzung agiler Softwareentwicklung zwischen beiden Branchen auf?

3. Wie können agile Methoden für die Anwendung in der Medizintechnik und Automobilindustrie für die Lösung der Probleme für die ausgewählten Prozesse angepasst werden?

3 Methoden

Um die erste Forschungsfrage, die Probleme der Anwendung von agiler Softwareentwicklung im Medizintechnik- und Automobiltechnikbereich, beantworten zu können, wurden zwei Schlüsselaufgaben definiert.

Der erste Teil beschäftigte sich mit einer umfassenden Analyse von den beiden agilen Softwareentwicklungsmethoden Scrum und Kanban (Abschnitt 1.3). Dabei wurde der Fokus der Recherche auf häufig verwendete Artefakte, Ereignisse und Rollen im Lebenszyklus von Scrum und auf die Hauptpraktiken von Kanban gelegt. Weiteres wurden der Automobiltechnikstandard Automotive SPICE 3.1, vor allem die Struktur des darin enthaltenen Prozessreferenzmodells, und die Software-Entwicklungs-Prozesse und -Aktivitäten der Medizintechniknorm EN 62304 analysiert (Abschnitt 1.4). Mit dem daraus gewonnenen Wissensstand wurden im zweiten Teil einerseits, die für Softwareentwicklung in der Medizintechnik relevante Norm EN 62304 und der Standard ASPICE 3.1 für Automobiltechnik auf Gemeinsamkeiten und Unterschiede untersucht (Abschnitt 3.1) und andererseits wurden mittels Gap-Analyse ausgewählte Prozesse/Aktivitäten auf Übereinstimmungen und Konflikte mit den Artefakten, Ereignissen und Rollen von Scrum und den Praktiken von Kanban analysiert (Abschnitt 3.2).

Zur Lösung der zweiten und dritten Forschungsfrage wurden Adaptionen für bestehende agile Praktiken und andere Praktiken bzw. Tools aus der Softwareentwicklung gesucht, um einerseits einen Vorschlag für die gemeinsamen Problemfelder, welche in Forschungsfrage zwei gefordert sind, und andererseits für alle separat ermittelten Lücken zwischen agiler Softwareentwicklung und EN 62304 bzw. ASPICE 3.1 zu entwickeln.

3.1 EN 62304 und Automotive SPICE 3.1 im Vergleich

Um eine Übersicht der Gemeinsamkeiten der beiden Normen zu erhalten und Prozesse für die folgende Gap-Analyse, basierend auf den Ergebnissen, auswählen zu können, wurden alle Software-Entwicklungs-Prozesse und -Aktivitäten von EN 62304 aus Abschnitt 1.4.2 und des „VDA Scopes“ von ASPICE 3.1 aus Abbildung 1.10 ausgewählt und gegenübergestellt.

Das Mapping zwischen den Aktivitäten aus EN 62304 und der Prozesse aus ASPICE wurde durch den inhaltlichen Vergleich der Aufgaben (EN62304) mit den BPs (ASPICE) bewerkstelligt. Aufgrund von unterschiedlicher Nomenklatur der beiden Normen wurden Aktivitäten aus EN 62304 mit Prozessen aus ASPICE verglichen.

Die Berechnung der Übereinstimmung zwischen den Prozessen in ASPICE und den Aktivitäten in EN 62304, wurde am Beispiel vom Prozess SWE.1 (Softwareanforderungsanalyse) und Aktivität § 5.2 (Analyse der SW-Anforderungen) dargestellt.

Dabei wird die EN 62304-Aktivität § 5.2 in ihre Aufgaben 5.2.1 - 5.2.6 aufgeteilt und ihnen je nach Gemeinsamkeiten mit den BPs aus SWE.1 ein Übereinstimmungsgrad zugeordnet (0 ... keine Übereinstimmung („Not achieved “), 1 ... teilweise Übereinstimmung („Partially achieved“), 2 ... mehrheitliche Übereinstimmung („Largely achieved“), 3 ... völlige Übereinstimmung („Fully achieved“), siehe Tab. 1).

Die Begriffe Partially, Largely und Fully werden in der Bewertung von Prozessen unter anderem auch in ASPICE 3.1 häufig verwendet.

Tab. 1: Inhaltlicher Vergleich zwischen den Aufgaben von EN 62304 Aktivität Analyse der Software Anforderungen (§ 5.2) - und den BPs des ASPICE Prozess SWE.1 - Softwareanforderungsanalyse. ÜG (Übereinstimmungsgrad)

Aufgabe aus EN 62304	Begründung	ÜG
5.2.1 Ableitung der SW-Anforderungen aus dem System-Anforderungen und Dokumentation	Die Aufgabe 5.2.1 aus EN 62304 wird in ASPICE in SWE.1 durch BP 1 vollständig erfüllt, da auch hier die SW-Anforderungen aus den System-Anforderungen abgeleitet und definiert werden müssen. (Fully)	3
5.2.2 Inhalt der SW-Anforderungen	Die Anforderungen von 5.2.2 werden in SWE.1 durch BP1 und BP4 nur großteils erfüllt, da in SWE.1 keine Anforderungen bzgl. Installation, Abnahme und Wartung berücksichtigt werden. Weiters wird eine Anwenderdokumentation nicht dezidiert vorgeschrieben. (Largely)	2
5.2.3 Einbeziehen von Risikokontroll-Maßnahmen in die SW-Anforderungen	Da Risikokontroll-Maßnahmen in SWE.1 nicht behandelt werden, wird auch die Aufgabe 5.2.3 nicht berücksichtigt.	0
5.2.4 Erneute Evaluation der Risikoanalyse	Die Risikoanalyse wird in SWE.1 nicht behandelt.	0
5.2.5 Aktualisierung von System-Anforderungen	Da in SWE.1 BP7 ebenfalls eine Konsistenz zwischen System- und SW-Anforderungen sichergestellt werden muss, ist 5.2.5 vollständig erfüllt. (Fully)	3
5.2.6 Verifizierung von SW-Anforderungen	Die Anforderungen von 5.2.6 werden in SWE.1 durch die BPs 3,5 und 6 fast vollständig erfüllt. Einzig die Implementierung von Anforderungen werden in SWE.1 nicht definiert. (Fully)	3

Im Anschluss daran, wurde mit Formel (1), aus der Anzahl der Aufgaben der Aktivität (N), der höchstmöglichen Bewertung (ÜG_{MAX}) und der Summe der Übereinstimmungsgrade ($\text{ÜG}_{5.2.i}$), die Übereinstimmung zwischen der Aktivität *Analyse der Softwareanforderungen*

(§ 5.2) und dem Prozess *Softwareanforderungsanalyse* (SWE.1) in Prozent ermittelt.

Die Ergebnisse der kompletten Analyse werden im Abschnitt 4.1 dargestellt.

$$\begin{aligned} \ddot{\mathrm{U}}_{5.2\text{-SWE.1}} &= \frac{1}{N \cdot \ddot{\mathrm{U}}\mathrm{G}_{MAX}} \cdot \sum_{i=1}^{N} \ddot{\mathrm{U}}\mathrm{G}_{5.2.i} \cdot 100 \\ &= \frac{1}{6 \cdot 3} \cdot \sum_{i=1}^{6} \ddot{\mathrm{U}}\mathrm{G}_{5.2.i} \cdot 100 \\ &= \frac{1}{18} \cdot 11 \cdot 100 \\ &\approx 60\% \end{aligned} \tag{1}$$

Der nächste Schritt bestand darin, dass aus dieser Analyse Prozess-Aktivitätenpaare mit Vergleichspotential ausgewählt und diese auf Übereinstimmungen und Konflikte mit den Artefakten, Ereignissen und Rollen von Scrum bzw. den Praktiken von Kanban verglichen wurden.

Dabei wurde aus jedem Prozess der Medizintechniknorm EN 62304 eine Aktivität ausgewählt, welche eine hohe Übereinstimmung mit dem entsprechenden Prozess aus dem Prozessreferenzmodells von Automotive SPICE 3.1 besitzt. Um eine geeignete Auswahl von Prozess-Aktivitätenpaare treffen zu können, flossen zusätzlich zu dieser Bewertung, der Analysebedarf dieser Paare und Erfahrungswerte in die Entscheidung ein. Aus diesem Grund gab es Ausnahmen, die in Abschnitt 5 näher erläutert werden. Dementsprechend wurden die Aktivitäten/Prozesse, Softwareanforderungsanalyse (§ 5.2/SWE.1), Änderungsantragsmanagement (§ 8.2/SUP.10) und Problemuntersuchung (§ 9.2/SUP.9) ausgewählt und analysiert.

3.2 Übereinstimmungen und Konflikte von ausgewählten Prozessen bzw. Aktivitäten mit agiler Softwareentwicklung

Dabei wurden die Anforderungen aus den BPs der ausgewählten Prozessen aus ASPICE bzw. aus den Aufgaben der Aktivitäten aus EN 62304, mittels Gap-Analyse separat auf mögliche Realisierung durch die Artefakte, Ereignisse und Rollen von Scrum und die Praktiken von Kanban analysiert. Um einen Überblick über den Abdeckungsgrad der Anforderungen durch diese agilen Praktiken zu erhalten, wurde den BPs und Aufgaben

prozentuelle Abdeckungen zugeordnet.

Die Berechnung der Abdeckung der BPs und Aufgaben wurde anhand der Aufgabe 8.2.4 aus EN 62304 exemplarisch dargestellt.

Dazu wurde die Aufgabe in ihre Anforderungen unterteilt und ihnen je nach Abdeckung durch agile Praktiken ein Abdeckungsgrad ($AG_{8.2.4_i}$) mit den Werten 0 (keine Abdeckung durch Scrum oder Kanban), 0.5 (teilweise Abdeckung) und 1 (vollständige Abdeckung), zugeordnet. Aus der daraus gewonnenen Summe der Abdeckungsgrade und der Anzahl der Anforderungen (N) konnte die prozentuelle Abdeckung der Aufgabe 8.2.4 gewonnen werden (siehe Formel (2)).

$$\begin{aligned} \text{Abdeckung}_{\,8.2.4} &= \frac{1}{N} \cdot \sum_{i=1}^{N} \text{AG}_{8.2.4_i} \cdot 100 \\ &= \frac{1}{3} \cdot (1 + 0 + 0) \cdot 100 \\ &\approx 33\% \end{aligned} \tag{2}$$

Um einen Überblick über die insgesamte Abdeckung in einem/r Prozess/Aktivität durch agile Praktiken zu bekommen, wurde aus den daraus gewonnenen Teilergebnissen $Abdeckung_{8.2.i}$ und der Anzahl der Aufgaben N der arithmetische Mittelwert gebildet (Formel (3)), welcher hierbei am Beispiel der Aktivität *Änderungskontrolle* (§ 8.2) dargelegt wurde (Tab. 6).

$$\begin{aligned} \text{Abdeckung}_{\,8.2} &= \frac{1}{N} \cdot \sum_{i=1}^{N} \text{Abdeckung}_{8.2.i} \\ &= \frac{1}{4} \cdot (50 + 100 + 0 + 33) \\ &\approx 46\% \end{aligned} \tag{3}$$

3.3 Entwicklung eines agilen Scrum- Lebenszyklus für ausgewählte Prozesse/Aktivitäten

Um einen agilen Lebenszyklus entwickeln zu können, der den Anforderungen aus den ausgewählten Prozessen und Aktivitäten der Normen entspricht, mussten Lösungen für die

aufgetretenen Lücken gefunden werden. Dabei wurde versucht agile Praktiken anzupassen, und andere Praktiken und Tools aus der Softwareentwicklung dafür zu verwenden. Nachdem Lösungen für die Probleme gefunden wurden, wurden alle Anforderungen aus den Normen, eine Beschreibung der Normkonformität und die dazugehörigen Artefakte/Praktiken, mit der die Normkonformität hergestellt werden kann, in Form einer Tabelle aufgelistet, um daraus im Anschluss den Lebenszyklus entwickeln zu können.

Als Beispiel für eine mögliche Anwendungsart von Scrum, welche die Anforderungen der Aktivität bzw. des Prozesses „Softwareanforderungsanalyse“ aus EN 62304 (§ 5.2) und ASPICE 3.1 (SWE.1) erfüllt, dient Tab. 9.

Der letzte Schritt bestand in der Generierung eines agilen Automobiltechnik- und Medizintechnikkonformen Scrum- Lebenszyklus, für die zuvor ausgewählten Prozesse/Aktivitäten. Dabei wurde ein agiler Standard - Scrum-Lebenszyklus, wie er in Abbildung 1.5 dargestellt ist, verwendet und durch die Auflistungen der Ergebnisse aus den zuvor durchgeführten Analysen (z.B. Tab. 9) abgeändert und ergänzt. Zur Erstellung des adaptierten Scrum Lebenszyklus, wurde das kostenlose Onlineprogramm *Cacoo: Online Diagram and Flowchart Software* (Nulab Inc., Fukuoka, Japan) verwendet.

Ein möglicher agiler Scrum-Lebenszyklus, der die Anforderungen der ausgewählten Aktivitäten/Prozesse Softwareanforderungsanalyse (§ 5.2/SWE.1), Änderungsantragsmanagement (§ 8.2/SUP.10) und Problemuntersuchung (§ 9.2/SUP.9) erfüllt, ist in den Abbildungen 4.1 und 4.2 ersichtlich und wie folgt aufgebaut:

1. Adaptierter Scrum-Lebenszyklus mit Angabe der verwendeten Artefakte, Rollen und Ereignisse aus Scrum bzw. Praktiken aus Kanban

2. Verweise auf die damit erfüllten BPs (ASPICE) und Aufgaben (EN 62304)

3. Angabe der Rollen, die an der Durchführung dieser Artefakte und Ereignisse beteiligt sind

4. Detailierte Darstellung von ausgewählten, verwendeten agilen und nicht-agilen Artefakten

4 Ergebnisse

4.1 EN62304 und Automotive SPICE 3.1 im Vergleich

Wie im Abschnitt 3.1 angegeben, wurden alle Aktivitäten aus EN62304 mit den Prozessen aus ASPICE 3.1 auf Übereinstimmung überprüft, um aus diesem inhaltlichen Vergleich (Tab. 2) drei Prozess-Aktivitätenpaare auszuwählen, die in weiteren Folge auf Übereinstimmung mit den agilen Softwareentwicklungsmethoden Scrum und Kanban verglichen wurden.

Aus den Ergebnissen dieser Analyse gingen vier unterschiedliche Übereinstimmungsbereiche hervor:

1. Vollständige/Teilweise Übereinstimmung ($> 0\%$) eines Prozess-Aktivitätenpaares aufgrund von inhaltlichen Gemeinsamkeiten zwischen den BPs und Aufgaben, wie z.B. zwischen der Aktivität *Design der SW-Architektur* (§ 5.3) und dem Prozess *Softwarearchitekturdesign* (SWE.2).

2. Keine inhaltliche Übereinstimmung zwischen den Aufgaben einer Aktivität aus EN 62304 und den BPs eines Prozesses aus Automotive SPICE 3.1, wie z.B. zwischen Aktivität *Unterrichtung beteiligter Stellen* (§ 9.3) und *Problemlösungsmanagement* (SUP.9).

3. Aktivitäten aus EN 62304 werden im Prozessreferenzmodell von Automotive SPICE 3.1 nicht behandelt, wie z.B. Aktivität *Risikokontroll-Maßnahmen* (§ 7.2)

4. Prozesse aus Automotive SPICE 3.1 werden in EN 62304 nicht berücksichtigt, wie z.B. *Qualitätssicherung* (SUP.1)

Die restlichen Teilergebnisse zur Berechnung der prozentuellen Übereinstimmungen befinden sich im Anhang.

Aus den Kapiteln mit Übereinstimmungen zwischen den beiden Normen (Software- Entwicklungsprozess, Software- Konfigurationsmanagementprozess und Problemlösungsprozess für Software aus EN 62304) wurde jeweils ein Prozess-Aktivitätenpaar für die weiteren Analysen ausgewählt (Tab. 2, hellgrün markiert).

Tab. 2: Inhaltlicher Vergleich zwischen den Aktivitäten aus EN 62304 und den Prozessen des VDA Scopes aus Automotive SPICE 3.1. Der Übereinstimmungsgrad wurde in Prozent angegeben und die ausgewählten Prozess-Aktivitätenpaare, für die nachfolgenden Analysen, hellgrün markiert.

Medizintechnik (EN 62304)			Automotive SPICE (VDA Scope)		Übereinstimmung
Prozess	Aktivitäten	Akt. ID	Prozess	Prozess ID	-
SW- Entwicklungsprozess	Planung der SW- Entwicklung	5.1	Project Management	MAN.3	20%
	Analyse der SW- Anforderungen	5.2	SW- Anforderungsanalyse	SWE.1	60%
	Design der SW- Architektur	5.3	SW- Architekturdesign	SWE.2	40%
	Detailliertes SW- Design	5.4	SW- Detailliertes Design & Einheitenkonstruktion	SWE.3	100%
	Implementierung und Verifizierung der SW- Einheiten	5.5	Verifizierung der SW- Einheit	SWE.4	60%
	SW- Integration & -Integrationsprüfung	5.6	SW- Integration & Integrationstest	SWE.5	60%
	Prüfung des SW- Systems	5.7	SW- Qualifizierungstest	SWE.6	50%
	SW- Freigabe	5.8	N/A	N/A	0%
SW- Risikomanagement prozess	Analyse von SW, die zu Gefährdungssituationen beiträgt	7.1	N/A	N/A	0%
	Risikokontroll- Maßnahmen	7.2	N/A	N/A	0%
	Verifizierung von Risikokontroll Maßnahmen	7.3	N/A	N/A	0%
	Risikomanagement von SW- Änderungen	7.4	N/A	N/A	0%
SW- Konfigurations management prozess	Identifizierung der Konfiguration	8.1	Konfigurationsmanagement	SUP.8	70%
	Änderungskontrolle	8.2	Änderungsantragsmanagement	SUP.10	80%
	Aufzeichnungen über den Status der Konfiguration	8.3	Konfigurationsmanagement	SUP.8	100%
Problemlösungs- Prozess für SW	Erstellung von Problemberichten	9.1	Problemlösungsmanagement	SUP.9	100%
	Untersuchung des Problems	9.2	Problemlösungsmanagement	SUP.9	100%
	Unterrichtung beteiligter Stellen	9.3	Problemlösungsmanagement	SUP.9	0%
	Anwendung des Änderungskontroll- Prozesses	9.4	Änderungsantragsmanagement	SUP.10	80%
	Aufbewahrung von Aufzeichnungen	9.5	Problemlösungsmanagement	SUP.9	100%
	Analyse von Problemen hinsichtlich Trends	9.6	Problemlösungsmanagement	SUP.9	100%
	Verifizierung des Lösung von SW- Problemen	9.7	Problemlösungsmanagement	SUP.9	100%
	Inhalt von Prüfungsdokumentation	9.8	Problemlösungsmanagement	SUP.9	0%
-	N/A	N/A	Qualitätssicherung	SUP.1	0%
-	N/A	N/A	Lieferanten- Monitoring	ACQ.4	0%
-	N/A	N/A	Anforderungserhebung	SYS.1	0%
-	N/A	N/A	System- Anforderungsanalyse	SYS.2	0%
-	N/A	N/A	System- Architekturdesign	SYS.3	0%
-	N/A	N/A	System- Integration & Integrationstest	SYS.4	0%
-	N/A	N/A	System- Qualifizierungstest	SYS.5	0%

4.2 Übereinstimmungen und Konflikte von ausgewählten Prozessen bzw. Aktivitäten mit agiler Softwareentwicklung

In diesem Abschnitt werden die Resultate aus den Gap-Analysen über die ausgewählten Prozess-Aktivitätenpaare Softwareanforderungsanalyse (§ 5.2/SWE.1), Änderungsantragsmanagement (§ 8.2/SUP.10) und Problemuntersuchung (§ 9.2/SUP.9) präsentiert.

4.2.1 Herausforderungen bei der agilen Analyse von Softwareanforderungen

Bei der ersten Untersuchung wurden der Prozess *Softwareanforderunganalyse (SWE.1)* aus Automotive SPICE 3.1 (Tab. 3) und die Aktivität *Analyse der Softwareanforderungen* (§ 5.2) aus EN 62304 (Abb. 4) mit den Artefakten, Ereignissen und Rollen von Scrum bzw. den Praktiken von Kanban verglichen.

Dabei wurde sowohl eine jeweilige Abdeckung zwischen den BPs bzw. Aufgaben und den agilen Praktiken, als auch eine insgesamte Abdeckung des Prozesses bzw. der Aktivität durch die agilen Praktiken ermittelt.

4.2.1.1 Softwareanforderungsanalyse (SWE.1)

Die Gap-Analyse zwischen den einzelnen BPs aus dem Prozess Softwareanforderungsanalyse (SWE.1) und den Möglichkeiten aus Scrum/Kanban zeigte, dass die BPs einerseits vollständig (z.B. BP1) oder teilweise (z.B. BP2) durch Artefakte, Ereignisse und Rollen aus Scrum, und den Praktiken aus Kanban abgedeckt werden können und andererseits deutliche Lücken zwischen den Anforderungen der Norm und den agilen Methoden bestehen (z.B. bei der Herstellung von bidirektionaler Traceability).

Der arithmetische Mittelwert der Abdeckung zwischen diesem Prozess und den beiden agilen Methoden Scrum und Kanban betrug 40%. Um eine 100-prozentige Abdeckung erreichen und somit alle Anforderungen aus dem Prozess erfüllen zu können, würden weitere Methoden aus der Softwareentwicklung oder Anpassungen in den bestehenden agilen Artefakten benötigt werden.

Die vollständige Analyse ist in Tab. 3 dargestellt.

4.2.1.2 Analyse der Softwareanforderungen (§ 5.2)

Durch die Gap-Analyse zwischen den Aufgaben aus der Aktivität *Analyse der Softwareanforderungen* (§ 5.2) und den Möglichkeiten aus den beiden agilen Methoden Scrum und Kanban konnte herausgefunden werden, dass diese Aufgaben aus der Norm wiederum vollständig (z.B. Aufgaben-ID 5.2.1: Ableitung der SW-Anforderungen aus Systemanforderungen und Dokumentation), teilweise (z.B. Aufgaben-ID 5.2.6: Verifizierung der SW-Anforderungen) oder überhaupt nicht, durch die Möglichkeiten aus Scrum und Kanban, eingehalten werden können.

Trotz einer mittleren Abdeckung von 67% und somit einem höheren Ergebnis im Vergleich zur Softwareanforderungsanalyse aus ASPICE 3.1, müssten auch in diesem Fall Lösungen für die Lücken zwischen § 5.2 und den agilen Möglichkeiten gefunden werden, um die Softwareanforderungen der Aktivität entsprechend analysieren zu können.

Die Ergebnisse dieser Analyse sind in Tab. 4 ersichtlich.

Tab. 3: Gap-Analyse zwischen dem Softwareanforderungsanalyse -Prozess (SWE.1) aus Automotive SPICE 3.1 und den Artefakten, Ereignissen, Rollen aus Scrum bzw. den Praktiken aus Kanban. Angabe der Abdeckungen der einzelnen BPs aus SWE.1 und einer insgesamten Abdeckung von SWE.1 durch die Möglichkeiten der agilen Methoden Scrum und Kanban in Prozent. Durch zusätzliche Aufführung der Lücken und Anwendungs-(Lösungs-) vorschläge im Kommentarfeld, kann die Angabe der Abdeckung nachempfunden werden. VV (Verbesserungsvorschlag), AG (Abdeckungsgrad)

Software Anforderungsanalyse (SWE.1)		Scrum Artefakte															Scrum Ereignisse						Scrum Rollen						Kanban Praktiken							Kommentar	Abdeckung
		Product Backlog	Release Backlog	Sprint Backlog	Produkt Inkrement	User Story	Impediment	Impediment Backlog	Taskboard	Sprint Burndown Chart	Velocity Chart	Sprint Ziel	Definition of Ready	Definition of done	Akzeptanzkriterien	Planning Poker	Release Planning	Sprint Planning	Daily Scrum	Backlog Refinement	Sprint Review	Sprint Retrospektive	Product Owner	Entwicklungsteam	Scrum Master	Quality Product Owner	Safety Product Owner	Agile Coach	Kanban Board	Work in Progress (WIP)-Limit	Swim Lanes	Akzeptanzkriterien	Metriken	tägliche Statusmeetings	Retrospektiv-Meeting		
ID	SPICE BPs																																			
BP1	Festlegen der Softwareanforderungen	x	x	x		x											x	x		x			x	x	x				x			x				-Kann mittels User Stories in Product/Release Backlog festgelegt werden bzw. im Sprint Planning Meeting ausgewählte Items im Sprint Backlog aufgenommen werden (AG = 1) -Changes mittels Backlog Refinement aufgefangen (AG = 1) - Im Kanbanboard können Anforderungen aufgenommen und mittels Swimlanes in Functional&NonFunctional Requirements aufgeteilt werden (AG = 1)	100%
BP2	Strukturieren der Softwareanforderungen	x				x														x			x	x	x				x			x				- Gruppierungsmöglichkeit nicht definiert (AG = 0) - Sortierung durch Product Backlog erfasst und bei etvaigen Changes im Product Backlog Refinement Meeting angepasst (AG = 1) - Kategorisierung durch Scrum nicht gegeben (AG = 0) - Priorisierung wird im Product Backlog vom PO und dem Entwicklungsteam durchgeführt bzw. im Backlog Refinement verändert (AG = 1)	50%
BP3	Analyse der Softwareanforderungen					x						x			x			x					x	x	x						x					- Gegenseitiges Widersprechen wird durch DoR Eintrag verhindert (AG = 1) - Technische Machbarkeit/Korrektheit kann mittels Definition of Ready in Scrum und durch Akzeptanzkriterien in Kanban überprüft werden. (AG = 1) - Risikoidentifikation nicht untersuchbar (AG = 0) - Auswirkungen auf Technische Auswirkungen nicht definiert (AG = 0) - Auswirkungen auf Kosten indirekt über Story Points/Planning Poker gegeben (AG = 1) - Auswirkungen auf Zeitplan mittels Planning Poker schätzbar (AG = 1)	67%
BP4	Bestimmung der Auswirkungen auf die Betriebsumgebung																																		Könnten im Product/Release Backlog enthalten sein, muss aber nicht. Also nicht genau definiert z.B. 8GB RAM notwendig (AG = 0) - In DoR überprüfen, ob es analysiert wurde. (VV)	0%
BP5	Entwicklung von Verifikationskriterien																																		- Verifikationskriterien könnten in DoR/DoD Checklisten in Scrum oder als Akzeptanzkriterien in Scrum & Kanban aufgenommen werden. Nicht genau definiert (VV) (AG = 0)	0%
BP6	Herstellen von birektionaler Traceability																																		-Traceability zw. Requirements ist in Scrum nicht vorgegeben (JIRA bietet Möglichkeit von Traceability) (AG = 0)	0%
BP7	Sicherstellen von Übereinstimmung																																		-Consistency zw. Requirements ist in Scrum nicht vorgegeben (JIRA bietet die Möglichkeit) (AG = 0)	0%
BP8	Kommunikation der bestätigten Softwareanforderungen	x	x	x		x						x					x	x					x	x	x										- Bestätigte Anforderungen (DoR erfüllt) werden im Product Backlog als Ready gekennzeichnet und können im nächsten Release/Sprint Planning Meeting vom Entwicklungsteam/PO ins Release/Sprint Backlog ausgewählt werden, wodurch sie auch kommuniziert werden (AG = 1)	100%
																																			MW:	40%

Tab. 4: Gap-Analyse zwischen der Aktivität „Analyse der Software-Anforderungen“ (§ 5.2) aus EN62304 und den Artefakten, Ereignissen, Rollen aus Scrum bzw. den Praktiken aus Kanban. Angabe der Abdeckungen der einzelnen Aufgaben aus § 5.2 und einer insgesamten Abdeckung von § 5.2 durch die Möglichkeiten der agilen Methoden Scrum und Kanban in Prozent. Durch zusätzliche Aufführung der Lücken und Anwendungs-(Lösungs-)vorschläge im Kommentarfeld, kann die Angabe der Abdeckung nachempfunden werden. VV (Verbesserungsvorschlag), AG (Abdeckungsgrad)

Analyse der Softwareanforderungen (§ 5.2)		Scrum Artefakte															Scrum Ereignisse						Scrum Rollen						Kanban Praktiken							Kommentar	Abdeckung
		Product Backlog	Release Backlog	Sprint Backlog	Produkt Inkrement	User Story	Impediment	Impediment Backlog	Taskboard	Sprint Burndown Chart	Velocity Chart	Sprint Ziel	Definition of Ready	Definition of done	Akzeptanzkriterien	Planning Poker	Release Planning	Sprint Planning	Daily Scrum	Backlog Refinement	Sprint Review	Sprint Retrospektive	Product Owner	Entwicklungsteam	Scrum Master	Quality Product Owner	Safety Product Owner	Agile Coach	Kanban Board	Work in Progress (WIP)-Limit	Swim Lanes	Akzeptanzkriterien	Metriken	tägliche Statusmeetings	Retrospektiv-Meeting		
Aufg.-ID	Aufgaben																																			
5.2.1	Ableitung der SW-Anforderungen aus Sys-Anforderungen und Dokumentation	x	x	x		x											x	x					x	x					x							- Abgeleitete Anforderungen werden im Product Backlog dargestellt. (AG = 1) - SW- & Sys- Anforderungen im Product Backlog (AG = 1)	100%
5.2.2	Anforderungen an Funktionalität & Leistungsfähigkeit	x				x																	x	x												Inhalte aus 5.2.2 könnten teilweise im Product Backlog als User Stories erfasst werden Zusätzliche Möglichkeit wäre die Erstellung einer Review Checklist, um den Inhalt des Product Backlogs auf Vollständigkeit zu überprüfen. (AG = 0.5)	50%
5.2.3	Risikokontrollmaßnahmen in SW-Anforderungen einbeziehen	x				x																														Wenn notwendig, werden Risikokontrollmaßnahmen als User Stories im Product Backlog aufgenommen. (AG = 1)	100%
5.2.4	Erneute Evaluation der Risikoanalyse																	x																		Risikoanalyse wird im Backlog Refinement Meeting beim Auftreten von neuen User Stories oder Änderungen erneut durchgeführt. (AG = 1)	100%
5.2.5	Aktualisieren der System-anforderungen																																			VV: Aktualisierung der zugehörigen Systemanforderungen könnte im Backlog Refinement Meeting durchgeführt werden => Verlinkung notwendig (AG = 0)	0%
5.2.6	Verifizierung der SW-Anforderungen	x	x	x		x							x	x			x	x													x					- keine Traceability zum System (AG = 0 - Gegenseitiges Widersprechen überprüfen (nicht definiert) (AG = 0) - Vermeidung von Mehrdeutigkeit mittels User Stories (AG = 1) - d) mittels DoR/DoD in Scrum oder Akzeptanzkriterien in Scrum & Kanban (AG = 1) - e) mittels ID der User Story im Product Backlog (AG = 1) - f) nicht definiert (AG = 0)	50%
																																			MW:	67%

4.2.2 Herausforderungen bei der agilen Analyse von Änderungen

Der zweite Bereich der mittels Gap-Analyse auf mögliche Umsetzung durch agile Methoden untersucht wurde, war die Änderungsanalyse. Dabei wurden der Prozess *Änderungsantragsmanagement (SUP.10)* aus Automotive SPICE 3.1 (Tab. 5) und die Aktivität *Änderungskontrolle (§ 8.2)* aus EN 62304 (Tab. 6) mit den Artefakten, Ereignissen und Rollen von Scrum bzw. den Praktiken von Kanban verglichen.

Wiederum wurde eine jeweilige Abdeckung zwischen den BPs bzw. Aufgaben und den agilen Praktiken, als auch eine generelle Abdeckung des Prozesses bzw. der Aktivität ermittelt.

4.2.2.1 Änderungsantragsmanagement (SUP.10)

Aus der Gap-Analyse zwischen den einzelnen BPs aus dem Prozess Änderungsantragsmanagement SUP.10 und den Möglichkeiten aus Scrum bzw. Kanban ging hervor, dass sich das Änderungsantragsmanagement aus Automotive SPICE 3.1 gut agil umsetzen lassen würde.

Bis auf die Ausnahme der fehlenden Möglichkeit der Herstellung von bidirektionaler Traceability zwischen Änderungen, Arbeitsprodukten und Problemen, konnten die BPs durch agile Praktiken nahezu vollständig abgedeckt werden, wodurch sich eine insgesamte Abdeckung von 75% ergab.

In Tab. 5 ist die vollständige Analyse dargestellt.

4.2.2.2 Änderungskontrolle (§ 8.2)

Nach Analyse der Lücken zwischen den Aufgaben aus der Änderungskontrolle § 8.2 und den Praktiken aus den agilen Methoden Scrum und Kanban (Tab. 6) konnte eine niedrige Abdeckung der Aufgaben dieser Aktivität festgestellt werden. Einzig die Implementierung von Änderungen könnte vollständig durch agile Praktiken umgesetzt werden.

Dementsprechend betrug der arithmetische Mittelwert der Abdeckung von § 8.2 aus EN 62304 nur 46%.

Tab. 5: Gap-Analyse zwischen dem Änderungsantragsmanagement- Prozess (SUP.10) aus Automotive SPICE 3.1 und den Artefakten, Ereignissen, Rollen aus Scrum bzw. den Praktiken aus Kanban. Angabe der Abdeckungen der einzelnen BPs aus SUP.10 und einer insgesamten Abdeckung von SUP.10 durch die Möglichkeiten der agilen Methoden Scrum und Kanban in Prozent. Durch zusätzliche Aufführung der Lücken und Anwendungs-(Lösungs-)vorschläge im Kommentarfeld, kann die Angabe der Abdeckung nachempfunden werden. AG (Abdeckungsgrad)

Änderungsantragsmanagement (SUP.10)		Scrum Artefakte														Scrum Ereignisse						Scrum Rollen						Kanban Praktiken										Kommentar	Abdeckung
		Product Backlog	Release Backlog	Sprint Backlog	Produkt Inkrement	User Story	Impediment	Impediment Backlog	Taskboard	Sprint Burndown Chart	Velocity Chart	Sprint Ziel	Definition of Ready	Definition of done	Akzeptanzkriterien	Planning Poker	Release Planning	Sprint Planning	Daily Scrum	Backlog Refinement	Sprint Review	Sprint Retrospektive	Product Owner	Entwicklungsteam	Scrum Master	Quality Product Owner	Safety Product Owner	Agile Coach	Kanban Board	Work in Progress-Limit	Swim Lane's	Akzeptanzkriterien	Metriken	tägliche Statusmeetings	Retrospektiv-Meeting		
ID	SPICE BPs																																					
BP1	Entwicklung einer Änderungsmanagement-strategie	x				x			x													x	x					x									- Änderungsmanagementstrategie in Scrum nicht explizit definiert (VV: Dokument - Change Management Plan; Erstellung und Vollständigkeit des Dokuments über DoR Eintrag überprüfbar) (AG = 0) - Change Request activities werden als User Stories im Product Backlog aufgenommen. (AG = 1) - Ein Statusmodell ist in Kanban über das Kanban Board möglich oder in Scrum über Taskboard (AG = 1) - Nicht klar in Scrum definiert: Analysekriterien in Form von Akzeptanzkriterien der User Story über DoR gefordert (AG = 0) - Verantwortlichkeiten durch Auswahl der Tasks im Taskboard/Kanbanboard geklärt (AG = 1)	60%
BP2	Identifizierung und Aufzeichnung von Änderungsanträgen					x																x	x														- Eindeutig identifizierbar durch ID der User Story (AG = 1) - User Story beschreibt und dokumentiert Änderungsauftrag (AG = 1) - Aufgrund der Formatierung der User Story "As a <user>, I want <some goal> so that <some reason> ist Initiator und Grund der Änderung enthalten (AG = 1)	100%
BP3	Aufzeichnen des Änderungsantragsstatus								x													x	x					x									Über Taskboard/Kanban Board wird der Status des Änderungsauftrages ersichtlich. (AG = 1)	100%
BP4	Analyse und Bewertung von Änderungsanträge					x								x								x	x														- Analyse der Abhängigkeiten zu anderen Änderungen/Arbeitsprodukten durch Akzeptanzkriterien gefordert aber die Verlinkung ist in Scrum/Kanban nicht definiert (VV: JIRA Links) (AG = 0) - Auswirkungen der Änderungen teilweise definiert (zeitlich... story points, technisch... nicht definiert) (AG = 1) - Die Kriterien zur Bestätigung der Implementierung werden in den Akzeptanzkriterien festgelegt. (AG = 1)	67%
BP5	Bestätigung von Änderungsanträgen vor der Implementierung	x				x						x							x			x	x														- Priorisierung erfolgt beim Backlog Refinement Meeting im Product Backlog und wird durch die Kennzeichnung als "Ready" im Product Backlog bestätigt. (AG = 1) - Genehmigung der Änderung teilweise in Scrum möglich (kleine Änderungen -> Backlog Refinement Meeting durch Setzen auf "Ready" ->; größere Änderung => Management dabei... Change Control Board (VV)) (AG = 0.5)	75%
BP6	Überprüfung der Implementierung der Änderungsanträge					x			x					x	x								x	x														Mittels DoD Einträge wird überprüft, ob alle Akzeptanzkriterien der Änderung erfüllt und alle Tasks (Aktivitäten der Änderung) durchgeführt wurden. Erst danach darf die User Story auf "Done" gesetzt werden. (AG = 1)	100%
BP7	Verfolgen der Änderungsanträge bis zum Abschluss			x					x									x		x			x	x														Mittels Taskboard/Kanbanboard können die Änderungen täglich verfolgt werden. Zusätzlich wird Daily Scrum Meeting und Sprint Review Meeting abgehalten. (AG = 1)	100%
BP8	Herstellen von birektionaler Traceability																																					In Scrum/Kanban nicht definiert. VV: Die Verwendung von einer Traceability Matrix (z.B. JIRA Tracability Matrix) wäre möglich. Req.(User story) -> Änderung (User story) -> Aktivität(Task)(AG = 0)	0%
																																						MW:	75%

Tab. 6: Gap-Analyse zwischen der Aktivität „Änderungskontrolle“ (§ 8.2) aus EN62304 und den Artefakten, Ereignissen, Rollen aus Scrum bzw. den Praktiken aus Kanban. Angabe der Abdeckungen der einzelnen Aufgaben aus § 8.2 und einer insgesamten Abdeckung von § 8.2 durch die Möglichkeiten der agilen Methoden Scrum und Kanban in Prozent. Durch zusätzliche Aufführung der Lücken und Anwendungs-(Lösungs-)vorschläge im Kommentarfeld, kann die Angabe der Abdeckung nachempfunden werden. VV (Verbesserungsvorschlag), AG (Abdeckungsgrad)

Änderungskontrolle (§ 8.2)		Scrum Artefakte														Scrum Ereignisse						Scrum Rollen						Kanban Praktiken								
		Product Backlog	Release Backlog	Sprint Backlog	Produkt Inkrement	User Story	Impediment	Impediment Backlog	Taskboard	Sprint Burndown Chart	Velocity Chart	Sprint Ziel	Definition of Ready	Definition of done	Akzeptanzkriterien	Planning Poker	Release Planning	Sprint Planning	Daily Scrum	Backlog Refinement	Sprint Review	Sprint Retrospektive	Product Owner	Entwicklungsteam	Scrum Master	Quality Product Owner	Safety Product Owner	Agile Coach	Kanban Board	Work in Progress (WIP)-Limit	Swim Lanes	Akzeptanzkriterien	Metriken	tägliche Statusmeetings	Retrospektiv-Meeting		
Aufg.-ID	**Aufgaben**																																		**Kommentar**	**Abdeckung**
8.2.1	Genehmigung von Änderungsanforderungen	x	x	x		x						x				x	x					x	x												Kleinere Änderungen werden im Product Backlog als "Ready" gekennzeichnet und können somit im nächsten Release/Sprint Planning Meeting in das Release/Sprint Backlog aufgenommen werden (somit genehmigt).Größere Änderungen könnten mit Hilfe eines Change Control Board von Management/Kunde genehmigt werden (VV) (AG = 0.5)	50%
8.2.2	Implementierung von Änderungen					x			x				x	x									x	x												Mittels DoD Einträge wird überprüft, ob alle Akzeptanzkriterien der Änderung erfüllt und alle Tasks (Aktivitäten der Änderung) durchgeführt wurden. Erst danach darf die User Story auf "Done" gesetzt werden. (AG = 1)	100%
8.2.3	Verifizierung von Änderungen																																		In Scrum nicht definiert. Für jede Änderung (User Story) könnten Akzeptanzkriterien aufgenommen werden, welche durch DoR gefordert werden und die Einhaltung in DoD überprüft werden. DoD: Alle Tasks der Änderungen in Status "Done". Alle Akzeptanzkriterien der Änderungen wurden erfüllt.(AG = 0)	0%
8.2.4	Bereitstellung von Mitteln für die Rückverfolgbarkeit von Änderungen					x						x											x	x												Nur Rückverfolgbarkeit zur Genehmigung in Scrum definiert => Kennzeichnung als "Ready"(AG = 1) Verbesserungsvorschlag: JIRA Traceability Matrix a) Änderungsanforderung <=>SW-Anforderung (AG = 0) b) Problembericht (User Story) <=> SW-Anforderung (AG = 0)	33%
																																			MW:	46%

4.2.3 Herausforderungen bei der agilen Analyse von Problemen

Um die Herausforderungen bei der agilen Analyse von Problemen zu untersuchen, wurden der Prozess *Problemlösungsmanagement* (SUP.9) aus Automotive SPICE 3.1 (Abb. 7) und die Aktivität *Untersuchung des Problems (§ 9.2)* aus EN 62304 (Abb. 8) mit den Artefakten, Ereignissen und Rollen von Scrum bzw. den Praktiken von Kanban verglichen.

Weiters wurde eine Ermittlung der jeweiligen Abdeckung zwischen den BPs bzw. Aufgaben und den agilen Praktiken, sowie einer insgesamten Abdeckung des Prozesses bzw. der Aktivität durch die Möglichkeiten der agilen Methoden durchgeführt.

4.2.3.1 Problemlösungsmanagement (SUP.9)

Mittels Gap-Analyse zwischen den einzelnen BPs aus dem Prozess Problemlösungsmanagement SUP.9 und den Möglichkeiten aus Scrum bzw. Kanban konnte erkannt werden, dass eine Erfüllung der Anforderungen aus diesem Prozess durch Scrum/Kanban nur durch erhebliche Anpassungen bzw. Verwendung von zusätzlichen Artefakten möglich wäre.

Da unter anderem die Genehmigung von dringenden Lösungsmaßnahmen (BP5) oder die Analyse von Problemtrends (BP9) in den beiden agilen Methoden nicht definiert sind.

Aufgrund von BPs wie der Aufzeichnung des Problemstatus (BP3), welche als Basiseigenschaft von Scrum oder Kanban anzusehen ist, konnte die Abdeckung auf 56% angehoben werden. Die restlichen Ergebnisse der Analyse können in Tab. 7 nachgelesen werden.

4.2.3.2 Untersuchung des Problems (§ 9.2)

Nach der Lückenanalyse zwischen den Aufgaben aus der Aktivität *Untersuchung des Problems § 9.2* und den Praktiken aus den agilen Methoden Scrum und Kanban (Tab. 8) konnte eine vollständig der Aktivität entsprechende Abdeckung festgestellt werden. Da unter anderem die Problemuntersuchung und Ursachenidentifikation (§ 9.2 a) mittels *DoD*-Eintrag und *Swim Lane*, und die Dokumentation der Untersuchungs- und Evaluationsergebnisse (§ 9.2 c) durch Verwendung eines *Impediment Backlogs* hundertprozentig erfüllt werden kann.

Dementsprechend betrug der arithmetische Mittelwert der Abdeckung von § 9.2 aus EN 62304 100%.

Tab. 7: Gap-Analyse zwischen dem Problemlösungsmanagement -Prozess (SUP.9) aus Automotive SPICE 3.1 und den Artefakten, Ereignissen, Rollen aus Scrum bzw. den Praktiken aus Kanban. Angabe der Abdeckungen der einzelnen BPs aus SUP.9 und einer insgesamten Abdeckung von SUP.9 durch die Möglichkeiten der agilen Methoden Scrum und Kanban in Prozent. Durch zusätzliche Aufführung der Lücken und Anwendungs-(Lösungs-)vorschläge im Kommentarfeld, kann die Angabe der Abdeckung nachempfunden werden. VV (Verbesserungsvorschlag), AG (Abdeckungsgrad)

Problemlösungs management (SUP.9)		Scrum Artefakte																Scrum Ereignisse						Scrum Rollen						Kanban Praktiken							Kommentar	Abdeckung
		Product Backlog	Release Backlog	Sprint Backlog	Produkt Inkrement	User Story	Impediment	Impediment Backlog	Taskboard	Sprint Burndown Chart	Velocity Chart	Sprint Ziel	Definition of Ready	Definition of done	Akzeptanzkriterien	Planning Poker	Release Planning	Sprint Planning	Daily Scrum	Backlog Refinement	Sprint Review	Sprint Retrospektive	Product Owner	Entwicklungsteam	Scrum Master	Quality Product Owner	Safety Product Owner	Agile Coach	Kanban Board	Limit Work in Progress (WIP)-	Swim Lanes	Akzeptanzkriterien	Meriken	tägliche Statusmeetings	Retrospektiv-Meeting		
ID	SPICE BPs																																					
BP1	Entwicklung einer Problemlösungs-managementstrategie	x		x		x	x	x	x									x				x	x	x			x		x						- Problemlösungsstrategie in Scrum nicht definiert (VV: Dokument - Problem management Plan; Erstellung und Vollständigkeit des Dokuments über DoR Eintrag überprüfbar) (AG = 0) - Probleme werden im Impediment Backlog aufgenommen und danach in Swim lane (Taskboard/Kanban board) übertragen; Bei notwendigen Änderungen werden entsprechende User Stories im Product Backlog angelegt und bei Bugs Impediments direkt in Swim Lane bearbeitet. (AG = 1) - Statusmodell ist in Kanban über Kanbanboard oder Scrum mittels Taskboard realisierbar. (AG = 1) - Warnmeldungen nur intern möglich (Daily Scrum Meeting) (extern VV: JIRA Notifications) (AG = 0.5) - Verantwortlichkeiten durch Auswahl der Tasks im Taskboard/Kanbanboard geklärt(AG = 1) - Dringende Probleme könnten in der Swim lane im Task-/Kanbanboard bearbeitet werden. (VV) (AG = 0	58%
BP2	Identifizierung und Aufzeichnung des Problems						x	x															x	x	x											- Eindeutig identifizierbar durch ID des Impediments (AG = 1) Impediments beschreiben und dokumentieren die Probleme - Traceability zw. Impediment, Änderung und der Problemursprungs-Anforderung (User Story) in Scrum nicht gegeben (Traceability durch JIRA Traceability Matrix möglich)(AG = 0)	50%
BP3	Aufzeichnung des Problemstatus						x		x														x	x				x								Über Taskboard/Kanban Board wird der Status der Probleme ersichtlich.(AG = 1)	100%
BP4	Diagnostizieren der Ursache und Ermittlung der Auswirkung des Problems						x	x	x					x									x	x	x			x		x						- Die Ursachen- und Auswirkungsanalyse der Impediments (im Status "In Analysis") wird durch DoD Eintrag gefordert und die Ergebnisse im Impediment dokumentiert - ebenfalls im DoD gefordert.(AG = 1) - Kategorisierung im Scrum nicht definiert (VV: Bug - Lösung direkt in Swim Lane, kein Bug - in Impediment dokumentiert, Änderung - Impediment auf "Done" setzen und Änderung (User Story) anlegen. & Zuordnung von JIRA Labels als Kategorien)(AG = 0)	50%
BP5	Genehmigung einer dringenden Lösungs-maßnahme																																			In Scrum/Kanban nicht direkt definiert! (AG = 0) VV:Genehmigung von dringenden Lösungsmaßnahmen für Impediments im Kanban/Taskboard werden mittels Kanban-Akzeptanzkriterien zwischen Status "In Analysis" und "In Work" bewerkstelligt. z.B. "PO hat die Aktivitäten bestätigt". Sind diese erfüllt, ist die Genehmigung erteilt und die Maßnahmen können im Status "In Work" entwickelt und implementiert werden.	0%
BP6	Warnmeldungen auslösen																	x					x	x												- intern möglich (Daily Scrum Meeting) (AG = 1) -extern: nicht möglich (AG = 0) VV:Warnmeldungen an betroffene Parteien mittels JIRA Notifications aussenden => Parteien als "Assignee" definieren	50%
BP7	Initierung einer Problemlösung						x		x					x									x	x				x		x						- Wenn eine Problemlösung notwendig ist (Bug) => Impediment direkt in Swim Lane bearbeitet, getestet und abgeschlossen: DoD fordert bei Bug Problemlösungsaktivitäten und Tests.(AG = 1) - Im Falle einer Änderung fordert DoD für das Impediment die Erstellung und Verlinkung einer Änderung (User Story) und die Kanban-Akzeptanzkriterien erlaubt Sprung von "In Work in "Done"(AG = 1)	100%
BP8	Verfolgung von Problemen bis zum Abschluss						x		x					x									x	x				x		x						- Der Status von Problemen wird mittels Task-/Kanbanboard in der Swimlane verfolgt. Der Status von Änderungen wird im Task-/Kanbanboard verfolgt. (AG = 1) - Die Autorisierung der Problemlösungsabnahme wird mittels DoD Eintrag gefordert. Im Falle eines Bugs, muss das Problem das Board erfolgreich durchlaufen haben. Änderungen: z.B. Änderung muss abgeschlossen sein, bevor Probelm geschlossen werden kann (AG = 1)	100%
BP9	Analyse von Problemtrends																																			In Scrum nicht definiert.(AG = 0) VV: Mittels Tools (z.B. JIRA) könnten Probleme (User Stories) gesammelt und mittels Gadgets (z.B. Charts, Tabellen) Trends erkannt werden. Diese Problem Trends könnten im Backlog Refinement Meeting analysiert und projektbezogene Lösungen entwickelt werden, die als User Story im Product Backlog aufgenommen werden.	0%
																																					MW:	56%

Tab. 8: Gap-Analyse zwischen der Aktivität „Untersuchung des Problems“ (§ 9.2) aus EN62304 und den Artefakten, Ereignissen, Rollen aus Scrum bzw. den Praktiken aus Kanban. Angabe der Abdeckungen der einzelnen Aufgaben aus § 9.2 und einer insgesamten Abdeckung von § 9.2 durch die Möglichkeiten der agilen Methoden Scrum und Kanban in Prozent. Durch zusätzliche Aufführung der Lücken und Anwendungs-(Lösungs-)vorschläge im Kommentarfeld, kann die Angabe der Abdeckung nachempfunden werden. AG (Abdeckungsgrad)

Untersuchung des Problems (§ 9.2)		Scrum Artefakte															Scrum Ereignisse						Scrum Rollen						Kanban Praktiken										
		Product Backlog	Release Backlog	Sprint Backlog	Produkt Inkrement	User Story	Impediment	Impediment Backlog	Taskboard	Sprint Burndown Chart	Velocity Chart	Sprint Ziel	Definition of Ready	Definition of done	Akzeptanzkriterien	Planning Poker	Release Planning	Sprint Planning	Daily Scrum	Backlog Refinement	Sprint Review	Sprint Retrospektive	Product Owner	Entwicklungsteam	Scrum Master	Quality Product Owner	Safety Product Owner	Agile Coach	Kanban Board	Work in Progress (WIP)-Limit	Swim Lanes	Akzeptanzkriterien	Metriken	tägliche Statusmeetings	Retrospektiv-Meeting		
Aufg.-ID	**Aufgaben**																																			**Kommentar**	**Abdeckung**
9.2 a)	Problem untersuchen und Ursachen identifizieren						x	x	x					x									x	x					x		x						Das Problem wird als Impediment im Impediment Backlog angelegt und in die Swim Lane im Kanbanboard übernommen. Die Analyse des Problems und Identifizierung der Ursachen wird in DoD gefordert und im Status "In Analysis" durchgeführt. (AG = 1)	100%
9.2 b)	Sicherheitsrelevanz des Problems evaluieren						x		x														x	x					x		x	x					Mittels Kanban-Akzeptanzkriterien von Status "In Analysis" zu "In Work" muss das Problem unter Verwendung des Software-Risikomanagement-Prozesses die Relevanz des Problems analysieren. (AG = 1)	100%
9.2 c)	Untersuchungs- und Evaluations-ergebnisse dokumentieren						x							x									x	x					x								Die Ergebnisse der Evaluierung und Untersuchung werden im Impediment dokumentiert (Ergebnisdokumentation durch DoD gefordert)(AG = 1)	100%
9.2 d)	Änderungs-anforderung gegebenenfalls erstellen						x		x					x									x	x					x		x						- Wenn eine Problemlösung notwendig ist (Bug) => Impediment direkt in Swim Lane bearbeitet, getestet und abgeschlossen: DoD fordert bei Bug Problemlösungsaktivitäten und Tests. (AG = 1) - Im Falle einer Änderung fordert DoD für das Impediment die Erstellung und Verlinkung einer Änderung (User Story) und die Kanban-Akzeptanzkriterien erlaubt Sprung von "In Work in "Done" (AG = 1) - Ist keine Änderung oder Problemlösung notwendig => Akzeptanzkriterium erlaubt direkten Übergang in den Status "Done" und DoD fordert Begründung. (AG = 1)	100%
																																				MW:	100%

4.3 Entwicklung eines agilen Scrum-Lebenszyklus für ausgewählte Prozesse/Aktivitäten

Dieses Kapitel beinhaltet einen Vorschlag zur Anwendung der agilen Methode Scrum, welcher die Anforderungen der ausgewählten Prozesse und Aktivitäten erfüllt und in folgende zwei Bereiche gegliedert wird.

- Auflistung der BPs und Aufgaben aus den jeweiligen Analysepaaren mit zugehörigen agilen Artefakten und nicht-agilen Praktiken zur Erfüllung der Anforderungen aus der Norm und Anwendung von Scrum. Weiters ist eine Beschreibung des Vorgangs zur Erfüllung der Norm mit ausgewählter Praktik enthalten.

- Ein daraus ermittelter, den Anforderungen der Norm entsprechender agiler Scrum-Lebenszyklus

4.3.1 Analyse von Softwareanforderungen (SWE.1/§ 5.2) - Lösungsvorschlag zur agilen Umsetzung der Anforderungen aus den Normen

In Tab. 9 wird eine mögliche agile Umsetzung der Anforderungen aus der *Softwareanforderungsanalyse* aus Automotive SPICE 3.1 und aus der *Analyse der Softwareanforderungen* aus EN 62304 dargestellt. Dabei kann man erkennen, dass bei diesem Prozess-Aktivitätenpaar einige gemeinsame Anforderungen aufgetreten sind, welche somit auch mit einer (agilen) Methode lösbar waren. Für die restlichen Anforderungen, welche nur in einer der beiden Normen vorkamen, wurden ebenfalls Lösungsvorschläge angegeben.

Um eine agile Umsetzung der Anforderungen aus dem Prozess Softwareanforderungsanalyse SWE.1 und der Aktivität Analyse der Softwareanforderungen § 5.2 gewährleisten zu können, wurden einige nicht-agile Methoden wie z.B. die zusätzliche Verwendung einer *Review Checklist* oder die Anwendung einer *Traceability Matrix* benötigt.

Tab. 9: Vorschlag zur SWE.1/§ 5.2 konformen Anwendung von Scrum und Kanban. Bestehend aus den BPs aus SWE.1 (ASPICE 3.1), den Aufgaben aus § 5.2 (EN 62304), einer Beschreibung der Normkonformität und den Artefakten/Praktiken aus Scrum bzw. Kanban, mit denen die Konformität hergestellt werden kann.

BP-ID	EN62304	Normkonformität	Artefakt/Praktik
BP1	5.2.2	Alle bereits bekannten Functional- & Non Functional Requirements werden im Initial Backlog Refinement Meeting als User Stories in das Product Backlog aufgenommen.	Initial Backlog Refinement Meeting/Product Backlog/ User Story
BP1	5.2.2/5.2.3	Neu aufgetretene Anforderungen werden im Backlog Refinement Meeting als User Stories aufgenommen oder bei Änderungen bereits vorhandene User Stories angepasst.	Backlog Refinement Meeting/User Story
BP1	5.2.1	Von Systemanforderungen und -architektur abgeleitete SW-Anforderungen werden als User Stories im Product Backlog aufgenommen und verwaltet.	User Story/Product Backlog
-	5.2.2/5.2.3	Die Inhalte der Anforderungen werden mittels Review Checklist im Initial Backlog Refinement Meeting auf Vollständigkeit im Product Backlog überprüft.	Review Checklist/Initial Backlog Refinement Meeting
-	5.2.2/5.2.3	Neu aufgetretene Anforderungen oder Änderungen werden im Backlog Refinement Meeting mittels Review Checklist überprüft	Backlog Refinement Meeting/ Review Checklist
-	5.2.2/5.2.3	Durch die Verwendung einer Review Checklist werden die vorausgesetzten Inhalte von Softwareanforderungen aus 5.2 und die Einbeziehung von Risikokontrollmaßnahmen gewährleistet.	Review Checklist
-	5.2.4	Die Risikoanalyse wird im Backlog Refinement Meeting beim Auftreten von neuen Anforderungen oder Änderungen erneut durchgeführt.	Backlog Refinement Meeting
-	5.2.5	Nach Änderung oder Aufnahme von Softwareanforderungen wird die Aktualisierung von zugehörigen Systemanforderungen im (Initial) Backlog Refinement Meeting durchgeführt.	(Initial) Backlog Refinement Meeting
BP2	-	Die Priorisierung bzw. Sortierung der Anforderungen wird im Product Backlog erfasst und im Backlog Refinement Meeting bei etwaigen Changes angepasst	Product Backlog/Backlog Refinement Meeting
BP2	-	Die Gruppierung in projektrelevante Gruppen wird mit dem Tool „JIRA“ und der Verwendung von JIRA Epics vollzogen.	JIRA Epics
BP2	-	Kategorisierung der Anforderungen basierend auf den relevanten Kriterien für das Projekt durch Setzen von JIRA Labels.	JIRA Labels
BP3	-	Mittels DoR Eintrag wird sichergestellt, dass die technische Machbarkeit und Korrektheit der Anforderungen gegeben sind.	Definition of Ready
BP3	5.2.6	Gegenseitiges Widersprechen der Anforderungen wird durch die Aufnahme eines DoR Eintrags verhindert.	Definition of Ready
BP3	-	Auswirkungen auf Kosten und zeitliche Auswirkungen der Anforderungen werden über Zuordnung von Story Points der User Stories ermittelt. Story Points entsprechen bestimmter Stundenanzahl. Stunden x Stundenlohn =>Kosten	User Story/Story Points
BP3	-	Ein DoR Eintrag stellt sicher, dass alle technischen Auswirkungen der Anforderungen analysiert wurden, eine Risikoidentifikation möglich ist und gegebenenfalls im Product Backlog aufgenommen wurden.	Definition of Ready
BP4	5.2.2	Anforderungen an Schnittstellen/Betriebsumgebung werden als User Story beschrieben und im Product Backlog aufgenommen.	User Story/Product Backlog
BP4	-	Mittels DoR Eintrag wird sichergestellt, dass Auswirkungen der Anforderungen auf Schnittstellen und der Betriebsumgebung analysiert und gegebenenfalls im Product Backlog aufgenommen wurden.	Definition of Ready
-	5.2.3	Alle angemessenen Risikokontrollmaßnahmen werden als User Story im Product Backlog erfasst.	User Story/Product Backlog
BP5	5.2.6	Mittels DoR Eintrag wird sichergestellt, dass Akzeptanzkriterien für die Verifikation der Anforderungen aufgenommen wurden.	Definition of Ready
BP5	5.2.6	Mittels DoD Eintrag wird sichergestellt, dass Akzeptanzkriterien für die Verifikation der Anforderungen erfüllt wurden.	Definition of Done
BP5	-	Jede User Story bekommt Akzeptanzkriterien zur Erfüllung der Verifikation der SW-Anforderung.	Akzeptanzkriterien
BP6	5.2.6	Traceability zwischen System- und Softwareanforderungen, zwischen Systemarchitektur und Softwareanforderungen wird durch die Verwendung einer Traceability Matrix und Issue Links in JIRA gewährleistet.	JIRA Traceability Matrix/ JIRA Issue Links
BP7	5.2.6	Mittels DoR Eintrag wird die Konsistenz zwischen System- und Softwareanforderungen überprüft.	Definition of Ready
BP7	-	Mittels DoR Eintrag wird die Konsistenz zwischen Systemarchitektur und Softwareanforderungen überprüft.	Definition of Ready
BP8	-	Updates von SW-Anforderungen werden im Backlog Refinement Meeting besprochen, im Product Backlog aufgenommen und somit auch in weiterer Folge über die Aufnahme im Release/Sprint Planning Meeting ins Release/Sprint Backlog mehrfach an das Team kommuniziert.	Backlog Refinement Meeting/Sprint Planning Meeting/Release Planning Meeting
BP8	-	Anforderungen werden durch die Kennzeichnung der Anforderungen im Product Backlog als "Ready"bestätigt und durch die Aufnahme in das Release Backlog im Release Planning Meeting an das Team kommuniziert.	Product Backlog/ Release Planning Meeting
-	5.2.6	Ein DoR Eintrag stellt sicher, dass die Konsistenz zwischen Softwareanforderungen und Risiko-beherrschung überprüft wurde.	Definition of Ready
-	5.2.6	Mehrdeutigkeit wird über die Formulierung der User Stories vermieden und können über die Festlegung von IDs eindeutig identifiziert werden.	User Story

4.3.2 Analyse von Änderungen (SUP.10/§ 8.2) - Lösungsvorschlag zur agilen Umsetzung der Anforderungen aus den Normen

In Tab. 10 ist ein möglicher Lösungsvorschlag zur Anwendung von Scrum ersichtlich, welcher die Anforderungen des Änderungsantragsmanagement-Prozess aus ASPICE 3.1 und der Aktivität *Änderungskontrolle* erfüllt. Wobei auch bei diesem Prozess-Aktivitätenpaar einige gemeinsame Anforderungen aufgetreten sind, welche wiederum mit einer (agilen) Methode lösbar waren. Allen restlichen einseitigen Anforderungen konnten ebenfalls Methoden zur Lösung zugeordnet werden.

Wie im ersten Paar wurden auch hier agile Artefakte angepasst und nicht-agile Methoden (z.B. *Change Control Board*) zur Herstellung der Konformität mit ASPICE 3.1 und EN 62304 verwendet.

Tab. 10: Vorschlag zur SUP.10/§ 8.2 konformen Anwendung von Scrum und Kanban. Bestehend aus den BPs aus SUP.10 (ASPICE 3.1), den Aufgaben aus § 8.2 (EN 62304), einer Beschreibung der Normkonformität und den Artefakten/Praktiken aus Scrum bzw. Kanban, mit denen die Konformität hergestellt werden kann.

BP-ID	EN62304	Normkonformität	Artefakt/Praktik
BP1	-	Die Änderungsanforderungsstrategie im Projekt wird über DoR Eintrag gefordert und auf Vollständigkeit überprüft, und mit der Erstellung eines Änderungsmanagementplans umgesetzt.	Definition of Ready/ Änderungsmanagementplan
BP1	-	Änderungsanträge können jederzeit als User Story im Product Backlog aufgenommen werden.	User Story/Product Backlog
BP1	-	Das Statusmodell wird über die Spalten des Taskboards in Scrum bzw. Kanbanboard in Kanban ermöglicht.	Taskboard/Kanbanboard
BP1	-	Analysekriterien werden in Form von Akzeptanzkriterien der jeweiligen User Story zugeordnet und über DoR-Eintrag gefordert.	Definition of Ready/ Akzeptanzkriterien
BP2	-	Änderungsanträge werden durch User Stories beschrieben/dokumentiert und können durch die Zuordnung von IDs eindeutig identifiziert werden.	User Story
BP2	-	Der Initiator und Grund der Änderung wird aufgrund der Formatierung der User Story gewährleistet. As a <user>, I want <some goal> so that <some reason>.	User Story
BP3	-	Der Status des Änderungsantrags wird über das Taskboard/Kanbanboard ersichtlich.	Taskboard/Kanbanboard
BP4	-	Die Analyse der Abhängigkeiten zu anderen Änderungen/Arbeitsprodukten werden durch Akzeptanzkriterien gefordert und die gegenseitige Verlinkung durch JIRA Issue Links bewerkstelligt (in DoR enthalten)	JIRA Issue Links/ Akzeptanzkriterien/ Definition of Ready
BP4	-	Auswirkungen auf Kosten und zeitliche Auswirkungen der Anforderungen werden über Zuordnung von Story Points der User Stories ermittelt. Story Points entsprechen bestimmter Stundenanzahl. Stunden x Stundenlohn => Kosten	User story/Story points
BP4	-	Die Kriterien zur Bestätigung der Implementierung werden in den Akzeptanzkriterien festgelegt.	Akzeptanzkriterien
BP5	8.2.1	Kleinere Änderungen werden durch deren Kennzeichnung im Product Backlog als "Ready" genehmigt und können erst damit beim nächsten Release/Sprint Planning Meeting im Release/Sprint Backlog aufgenommen werden.	Product Backlog/ User Story
BP5	-	Die Priorisierung der Änderungen im Product Backlog erfolgt im Backlog Refinement Meeting.	Product Backlog/ Backlog Refinement Meeting
BP5	8.2.1	Größere Änderungen könnten mit Hilfe eines Change Control Board von Management/Kunde genehmigt werden und dadurch das Setzen auf „Ready“ erlaubt werden.	Change Control Board/ Definition of Ready
BP6	8.2.2	Mittels DoD Eintrag wird überprüft ob die Akzeptanzkriterien der Änderung erfüllt und alle Tasks (Aktivitäten der Änderung) durchgeführt wurden.	Akzeptanzkriterien/ Taskboard/ Definition of Done
BP7	-	Mittels Taskboard/Kanbanboard können die Änderungen täglich verfolgt werden. Zusätzlich wird Daily Scrum Meeting und Sprint Review Meeting abgehalten	Taskboard/Kanbanboard
-	8.2.3	Mittels DoR Eintrag wird sichergestellt, dass Akzeptanzkriterien für die Verifizierung der Änderungen aufgenommen wurden.	Definition of Ready
-	8.2.3	Mittels DoD Eintrag wird sichergestellt, dass Akzeptanzkriterien für die Verifizierung der Änderungen erfüllt wurden.	Definition of Done
-	8.2.4	Traceability zur Genehmigung der Änderungen durch die Kennzeichnung einer User Story als "Ready" gewährleistet.	User Story
BP8	8.2.4	Bidirektionale Traceability zwischen Änderungsanträgen und Arbeitsprodukten, oder zwischen Änderungsanträgen und Problemen wird durch die Verwendung einer Traceability Matrix und Issue Links in JIRA gewährleistet.	JIRA Traceability Matrix/ JIRA Issue Links

4.3.3 Analyse von Problemen (SUP.9/§ 9.2) - Lösungsvorschlag zur agilen Umsetzung der Anforderungen aus den Normen

In Tab. 11 wird ein Lösungsvorschlag zur agilen Umsetzung der Anforderungen aus dem Prozess SUP.9 (Problemlösungsmanagement) aus Automotive SPICE 3.1 und der Aktivität § 9.2 (Untersuchung des Problems) aus EN 62304 gezeigt. Wie für die beiden anderen Prozess-Aktivitätenpaare konnten auch in diesem Fall alle Anforderungen aus den Standards gelöst werden.

Dazu waren Anpassungen der agilen Artefakte und eine Zuhilfenahme von nicht-agilen Methoden, wie z.B. die Verwendung von *JIRA Labels* zur Kategorisierung der Probleme oder *JIRA Gadgets* zur Trendanalyse, notwendig.

Tab. 11: Vorschlag zur SUP.9/§ 9.2 konformen Anwendung von Scrum und Kanban. Bestehend aus den BPs aus SUP.9 (ASPICE 3.1), den Aufgaben aus § 9.2 (EN 62304), einer Beschreibung der Normkonformität und den Artefakten/Praktiken aus Scrum bzw. Kanban, mit denen die Konformität hergestellt werden kann.

BP-ID	EN62304	Normkonformität	Artefakt/Praktik
BP1	-	Die Problemlösungsstrategie im Projekt wird über DoR Eintrag gefordert und auf Vollständigkeit überprüft, und mit der Erstellung eines Problemmanagementplans umgesetzt.	Definition of Ready/ Problemmanagementplan
BP1	9.2 a)	Probleme werden im Impediment Backlog als Impediments aufgenommen und danach in die Swim Lane des Kanbanboards übertragen.	Impediment Backlog/Swim Lane/ Kanbanboard
BP1	-	Sind aufgrund des Problems Änderungen notwendig, so werden diese als User Stories im Product Backlog angelegt.	User Story
BP1	-	Das Statusmodell wird über das Taskboards in Scrum bzw. Kanbanboard in Kanban implementiert.	Taskboard/Kanbanboard
BP1	-	Warnmeldungen sind intern über Daily Scrum Meetings realisierbar und werden extern über sogenannte "Notifications"mit dem Tool JIRA an betroffene Parteien ausgesendet werden.	Daily Scrum Meeting/ JIRA Notifications
BP1	-	Probleme(Impediments) mit hoher Dringlichkeit werden direkt in der Swim Lane des Kanbanboards bearbeitet.	Impediment/Swim Lane
BP2	-	Probleme (Impediments) können durch die Zuweisung von IDs eindeutig identifiziert werden, welche durch Kanban-Akzeptanzkriterien zwischen „Sprint Backlog“ und „In Analysis“ gefordert werden.	Impediment/Kanban-Akzeptanzkriterien
BP2	-	Probleme werden im Impediment beschrieben und dokumentiert.	Impediment
BP2	-	Traceability zwischen Impediment, Änderung und der Problemursprungs- Anforderung (User Story) wird mittels JIRA Traceability Matrix gewährleistet.	JIRA Traceability Matrix
BP3	-	Über Taskboard in Scrum bzw. Kanbanboard in Kanban ist der aktuelle Status der Probleme ersichtlich.	Taskboard/Kanbanboard
BP4	9.2 a)	DoD- Eintrag fordert die Analyse der Ursachen und Auswirkungen des Problems, welche im Status „In Analysis“ im Kanbanboard durchgeführt wird.	Definition of Done
BP4	9.2 c)	DoD- Eintrag fordert die Dokumentation der Ursachen und Auswirkungen des Problems im Impediment.	Definition of Done/ Impediment
BP4	-	Die Kategorisierung der Probleme kann durch Setzen von JIRA Labels bewerkstelligt werden ("Bug", Änderung, "kein Problem")	JIRA Labels
BP5	-	Die Genehmigung von dringenden Lösungsmaßnahmen werden mittels Kanban-Akzeptanzkriterien zwischen Status „In Analysis“ und „In Work“ realisiert. z.B. „PO hat die Aktivität bestätigt?“	Kanban-Akzeptanzkriterien
-	9.2 b)	Kanban-Akzeptanzkriterien von Status „In Analysis“ zu „In Work“ fordern, dass das Problem unter Verwendung des Software-Risikomanagement-Prozess auf Relevanz analysiert wird.	Kanban-Akzeptanzkriterien
BP6	-	Interne Warnmeldungen sind über die Abhaltung von Daily Scrum Meetings kommunizierbar. Externe Warnmeldungen werden über JIRA Notifications an betroffene Parteien ausgesendet (Parteien als „Assignee “ definieren).	Daily Scrum Meeting/ JIRA Notifications
BP7/ BP4	9.2 d)	Wenn keine Lösungsmaßnahme für das Problem notwendig ist, wird dies im Impediment vermerkt, mittels DoD Eintrag auf Vorhandensein überprüft und Abschluss im Kanbanboard wird ermöglicht.	Impediment/Definition of Done/Kanbanboard
BP7/ BP4	9.2 d)	Mittels DoD Eintrag wird überprüft, ob bei notwendigen Problemlösungsaktivitäten das Impediment die Swim Lane durchlaufen hat (analysiert, implementiert, getestet, geschlossen).	Definition of Done/ Swim Lane
BP7/ BP4	9.2 d)	Im Falle einer notwendigen Änderung fordert DoD die Erstellung und Verlinkung einer Änderung (User Story). Kanban-Akzeptanzkriterien erlauben den Statuswechsel auf "Done"	Definition of Done/ User Story/Kanban-Akzeptanzkrierien
BP8	-	Mittels Task-/Kanbanboard wird der Status von Problemen (Swim Lane) und Änderungen verfolgt.	Taskboard/Kanbanboard
BP8	-	Mittels DoD Eintrag wird eine Bestätigung für den Abschluss des Problems vom PO gefordert, bevor es auf "Done"gesetzt werden kann.	Definition of Done
BP9	-	Mittels JIRA Gadgets (Tabellen, Charts, Filter,...) könnten, für die gesammelten Probleme (User Stories), Trends erkannt werden.	JIRA Gadgets
BP9	-	Problem Trends aus JIRA könnten im Backlog Refinement Meeting analysiert und projektbezogene Lösungen als User Stories aufgenommen werden.	Backlog Refinement Meeting/ User Story

4.3.4 Scrum-Lebenszyklus zur Erfüllung der Anforderungen aus den ausgewählten Aktivitäten aus EN 62304 und Prozessen aus ASPICE 3.1

In den beiden Abb. 4.1 und 4.2 ist ein Vorschlag eines angepassten Scrum- Lebenszykluses für die Automobil- und Medizintechnikkonforme Anwendung ersichtlich. Dabei wurden Artefakte und Ereignisse aus Scrum, Praktiken aus Kanban und nicht-agile Softwareentwicklungspraktiken für die Entwicklung verwendet. Die beiden folgenden Abbildungen

sind in fünf Bereiche - Lebenszyklus (1), Angabe der entsprechenden BPs (2)/Aufgaben (3), teilnehmende Rollen (4) und einer detailierten Auflistung von einzelnen Praktiken (5) - aufgebaut.

Der erste Teil des Lösungsvorschlages (Abb. 4.1) dient zur Analyse und Dokumentation der Anforderungen eines Projektes, welche in den beiden Normen durch die Aktivität *Analyse der Softwareanforderungen § 5.2* und dem Prozess *Softwareanforderungsanalyse SWE.1* abgedeckt sind.

Im ersten Abschnitt (1) von Abb. 4.1 ist der Scrum Lebenszyklus selbst, der aus den agilen Standardartefakten *Product Backlog*, *Release Backlog* und dem Sprint, welcher in *Taskboard*, *DoR* und *DoD* unterteilt ist, besteht. Weiters wurden die Scrum-Ereignisse *(Initial) Backlog Refinement Meeting*, *Release Planning Meeting*, *Sprint Planning Meeting*, *Daily Scrum Meeting*, *Sprint Retrospective Meeting* und *Sprint Review Meeting* und weitere nicht-agile Methoden, wie z.B. eine *Review Checklist*, für die Umsetzung der Anforderungen aus den Normen vorgeschlagen. Um die Aktivität § 5.2 bzw. den Prozess SWE.1 zu erfüllen waren detailierte Anpassungen der *Review Checklist* und der *DoR-/DoD-* Checklisten erforderlich. Am Ende jedes Sprints werden potentielle Softwareanforderungen geliefert.

In Abschnitt (2) wurden den jeweiligen Artefakten, Ereignissen und Methoden des Lebenszyklus die dadurch (teilweise) eingehaltenen BPs aus der Softwareanforderungsanalyse (SWE.1) zugeordnet, wie z.B. durch das *Release Backlog 1* in Kombination mit dem *Release Planning Meeting* Anforderungen aus BP8 erfüllt werden könnten. Diese Zuordnung wurde für die Aufgaben von § 5.2 aus EN 62304 in gleicher Art und Weise in Abschnitt (3) dargestellt, wodurch unter anderem erkennbar wird, dass die Aufgabe 5.2.6 (Verifizierung der SW-Anforderungen) zum einen Teil im Sprint und zum anderen Teil im *Product Backlog* durchgeführt werden könnte.

Der vierte Abschnitt veranschaulicht, dass bei der agilen Umsetzung der Softwareanforderungsanalyse zumindest die drei Rollen *Product Owner*, *Scrum Master* und das *Entwicklungsteam* teilnehmen sollten.

Im letzten Abschnitt (5) des Lösungsvorschlages wurde eine mögliche *Review Checklist* und die beiden angepassten agilen Checklisten *DoR* und *DoD* mit allen darin notwendigen Anforderungen aus dem Prozess-Aktivitätenpaar dargestellt.

Nach der ersten erfolgreichen Entwicklung von Softwareanforderungen für das Projekt

kann der zweite Teil des Lösungsvorschlages (Abb. 4.2) zur Analyse und Dokumentation von Problemen - aus SUP.9 (Problemlösungsmanagement) und § 9.2 (Untersuchung des Problems) - und Änderungen - SUP.10 (Änderungsantragsmanagement) und § 8.2 (Änderungskontrolle) angewandt werden. Im Großen und Ganzen wurden dazu die gleichen Hauptartefakte und Ereignisse aus Scrum, wie im ersten Teil des Lebenszyklus, verwendet. Zusätzlich dazu, mussten weitere agile Artefakte, wie z.B. das *Impediment Backlog* oder das *Kanbanboard*, und nicht-agile Methoden (z.B. *Change Control Board*) implementiert werden und Artefakte wie *DoR* oder *DoD* angepasst werden, um den Anforderungen der Normen zu entsprechen. Diese sind wiederum in ersten Abschnitt von Abb. 4.2 strukturiert dargestellt.

In den Abschnitten (2) und (3) sind nun zusätzlich zur Softwareanforderungsanalyse auch BPs und Aufgaben aus den beiden anderen Prozess-Aktivitätenpaaren den jeweiligen Artefakten zugeordnet. Im vierten Abschnitt sind wiederum die teilnehmenden Rollen angeführt und in Abschnitt (5) die zur Erfüllung der Anforderungen vorgeschlagenen agilen Methoden *Kanbanboard*, *DoR* und *DoD* detailiert dargestellt.

Eine genauere Beschreibung und die Art der Umsetzung des Lebenszyklus, um die Anforderungen aus den ausgewählten Prozessen/Aktivitäten zu erfüllen, können in den Tab. 9 - 11 in Kombination mit den beiden Abb. 4.1 und 4.2 entnommen werden.

Durch Anwendung dieses Lebenszyklus kommt es am Ende jedes Sprints zu einem potentiell lieferbaren Produktinkrement, in dem die Analyse von Anforderungen, Problemen und Änderungen EN 62304 und Automotive SPICE 3.1 konform abgewickelt wurde.

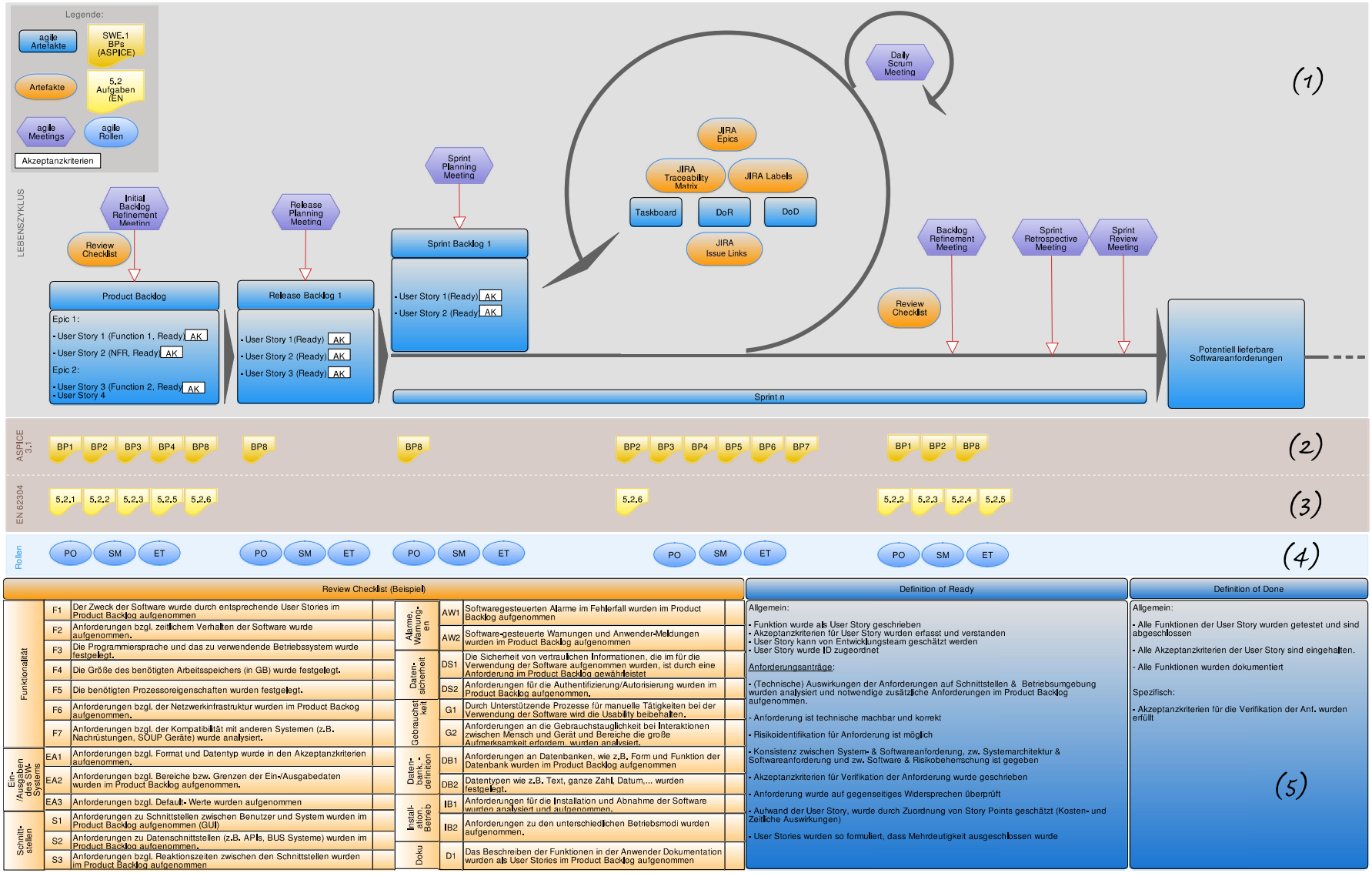


Abb. 4.1: Darstellung des ersten Teils eines Lösungsvorschlages zur agilen normkonformen Umsetzung der ausgewählten Prozesse/Aktivitäten (SWE.1/§ 5.2, SUP.10/§ 8.2 und SUP.9/§ 9.2) in Form eines Scrum-Lebenszyklus mit zusätzlicher Anwendung von Kanban Praktiken. Dieser Teil beinhaltet ausschließlich die Analyse von Softwareanforderungen. Die Abbildung ist in folgende vier Bereiche unterteilt: Lebenszyklus mit Legende, Angabe der damit erfüllten Aufgaben und BPs, teilnehmende Rollen und detailierte Auflistung der angewandten Artefakte *Review Checklist*, *DoR* und *DoD*.

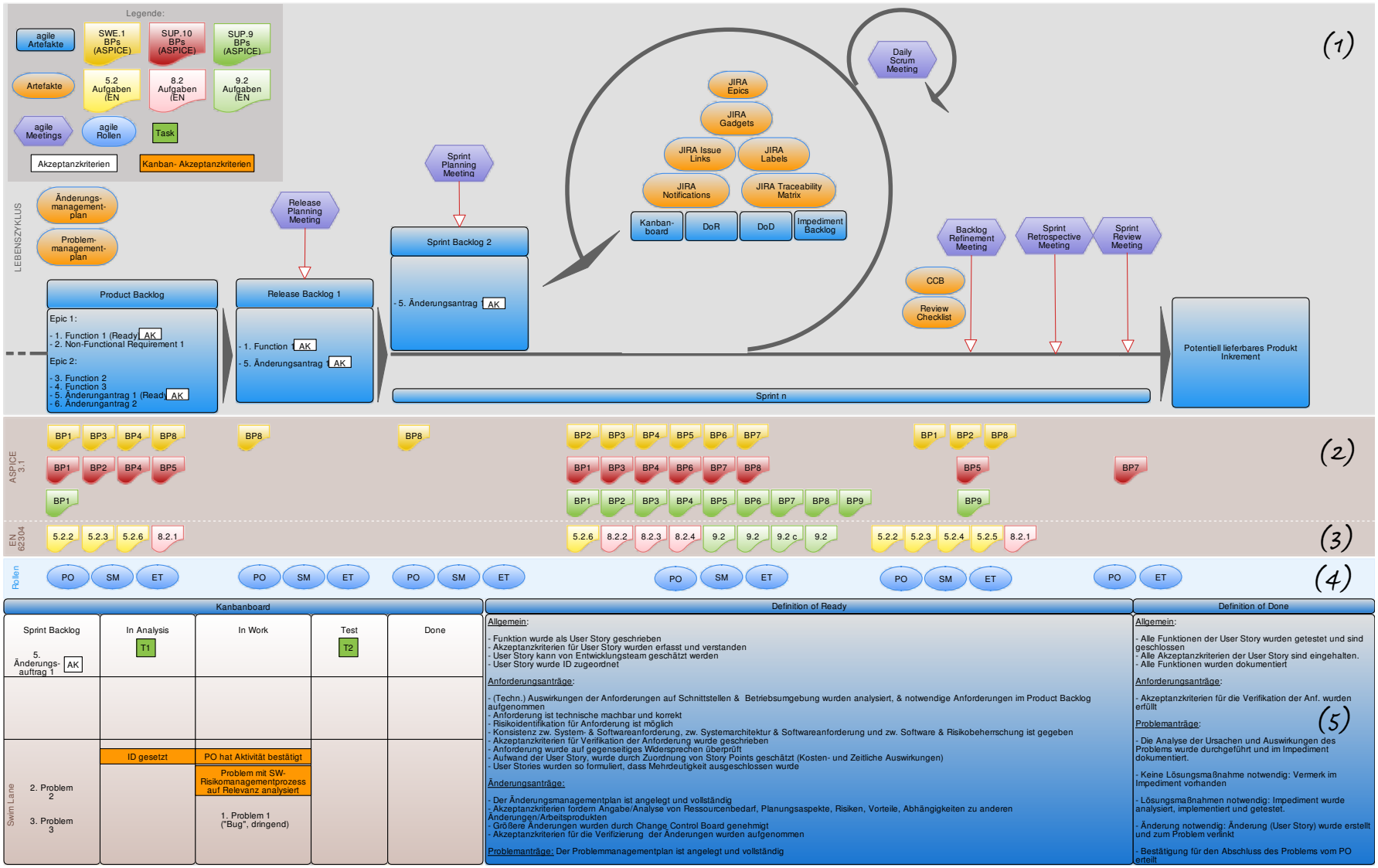


Abb. 4.2: Abbildung des zweiten Teils eines Lösungsvorschlages zur agilen normkonformen Umsetzung der ausgewählten Prozesse/Aktivitäten (SWE.1/§ 5.2, SUP.10/§ 8.2 und SUP.9/§ 9.2) in Form eines Scrum-Lebenszyklus mit zusätzlicher Anwendung von Kanban Praktiken. Dabei wird zusätzlich zur Analyse der Änderungen und Problemen, auch der Teil der Softwareanforderungen behandelt, der bei neuen Anforderungen oder Änderungen der Anforderungen zum Tragen kommt. Die Abbildung ist in folgende vier Bereiche unterteilt: Lebenszyklus mit Legende, Auflistung der damit erfüllten Aufgaben und BPs, teilnehmende Rollen und eine detailierte Angabe der Artefakte *Kanbanboard*, *DoR* und *DoD*.

5 Diskussion

5.1 EN 62304 und Automotive SPICE 3.1 im Vergleich

In Tab. 2 wird das Ergebnis des inhaltlichen Vergleichs zwischen ähnlichen Prozess-Aktivitätenpaare aus EN 62304 und ASPICE 3.1 präsentiert, um daraus Paare mit einer möglichst hohen Übereinstimmung für die weiteren Analysen auswählen zu können. Da das Ziel dieser Masterarbeit einen den Anforderungen aus EN 62304 und ASPICE 3.1 entsprechenden Lebenszyklus beinhaltet, und sich davon eine große Anzahl von gemeinsamen Herausforderungen ergeben sollten, wurde diese Vorgehensweise auserkoren. Wäre nun ein Prozess-Aktivitätenpaar mit einer geringen Übereinstimmung ausgewählt worden, dann hätten sich die Anforderungen weit voneinander unterschieden und eine gemeinsame Untersuchung wäre nicht sinnvoll gewesen. Deshalb wurden Paare mit fehlender Übereinstimmung (z.B. Software- Risikomanagementprozess oder Lieferantenmonitoring) in der Betrachtung für weitere Analysen außer Acht gelassen.

Im Nachhinein wurde erkannt, dass die berechnete Übereinstimmung zwischen den Paaren nur teilweise zutrifft, da sich im Laufe der Analysen weitere Herausforderungen ergeben haben, die auf den ersten Blick noch nicht ersichtlich waren und erst durch die folgenden Gap-Analysen zum Vorschein kamen.

Wie ist Tab. 2 ersichtlich, gab es bei der Auswahl der Paare mit der höchsten Übereinstimmung, Ausnahmen. Dies liegt daran, dass zusätzlich zur Übereinstimmung, Erfahrungswerte oder Analysebedarf in die Entscheidung eingeflossen sind.

Aus diesem Grund wurde aus dem ersten Kapitel „Softwareentwicklungsprozess“ nicht das detailierte Softwaredesign mit einer Übereinstimmung von 100% sondern die Analyse der Softwareanforderungen mit 60% ausgewählt. In diesem Fall spielt einerseits die Analyse der Softwareanforderungen bei Softwareprojekten im Allgemeinen eine tragende Rolle, womit auch das Interesse an der Untersuchung groß sei, und andererseits ist die Entwicklungsumgebung beim detailierten Softwaredesign zwischen den beiden Bereichen unterschiedlich (Modellbasiert, C).

Im zweiten Kapitel „Software- Konfigurationsmanagement“ wurde das Prozess- Aktivitätenpaar mit der zweithöchsten Übereinstimmung (Änderungskontrolle § 8.2/Änderungsantragsmanagement SUP.10) ausgewählt, da es bei der Aktivität „Aufzeichnungen über den

Status der Konfiguration“ (§ 8.3), mit einer Übereinstimmung von 100% zu Konfigurationsmanagement (SUP.8), nur eine Anforderung zu erfüllen gab, wodurch eine umfassende Analyse nicht notwendig war.

5.2 Übereinstimmungen und Konflikte der ausgewählten Prozesse/Akitiväten mit agiler Softwareentwicklung

Durch die Gap-Analysen zwischen den Anforderungen aus der Softwareanforderungsanalyse (SWE.1/§ 5.2) und den Praktiken von Scrum und Kanban wurde herausgefunden, dass eine Abdeckung von 40% bzw. 67% erreichbar sein kann. Dies ist großteils auf die fehlende Konsistenz und Traceability zwischen Softwareanforderungen zurückzuführen und findet auch Deckung in den Ergebnissen aus anderen Studien [33].

Bei den Analysen der Anforderungen aus der Änderungsanalyse (SUP.10/§ 8.2) konnte herausgefunden werden, dass einerseits die fehlende Möglichkeit der Verifizierung von Änderungen und andererseits wiederum die Traceability eine Abdeckungsergebnis von 75% und 46% verursacht. Dabei muss erwähnt werden, dass die Anzahl an Aufgaben aus der Aktivität § 8.2 im Vergleich zu den BPs aus SUP.10 deutlich niedriger ist und daher in die Betrachtung der Ergebnisse einbezogen werden muss.

Die beiden letzten Gap-Analysen des Prozesses über Problemlösungsmanagement (SUP.9), aus ASPICE 3.1, und der Aktivität „Untersuchung des Problems“ (§ 9.2), aus EN 62304, ergaben Abdeckungen von 56% bzw. 100%. Das deutlich schlechtere Ergebnis des Problemlösungs-management- Prozesses liegt einerseits daran, dass die Genehmigung von dringenden Lösungsmaßnahmen (BP5) und die Analyse von Problemtrends (BP 9) in Scrum und Kanban nicht behandelt werden, was auch in [33] herausgefunden wurde. Andererseits ist die Anzahl der Aufgaben aus § 9.2 wiederum deutlich geringer im Vergleich zu SUP.9 aus ASPICE 3.1 und es muss erwähnt werden, dass die Aktivität § 9.2 nur einem Teil der Problemlösungsanalyse in EN 62304 enstpricht.

Wie schon mehrfach angesprochen, wurden bei allen drei Bereichen jeweils Prozess- Aktivitätenpaare mittels Gap-Analysen mit den agilen Praktiken verglichen. Die Paarbildung zwischen Prozess und Aktivität baut auf der Tatsache auf, dass die Analyse der Anforderungen in ASPICE 3.1 im Prozess SWE.1 und in EN 62304 in der Aktivität § 5.2 geregelt wird. Um in der Masterarbeit ein einheitliches Konzept zu verfolgen wurde dieses Verfahren auch bei den beiden anderen Bereichen eingehalten.

Im Bereich Problemlösungs-Prozess für SW beinhaltet die analysierte Aktivität § 9.2 (Untersuchung des Problems) im Gegensatz zum Analysepartner SUP.9 nur einen Teil des Problemlösungsmanagements. Aus diesem Grund wäre ein Vergleich zwischen den Prozessen *Problemlösungsmanagement* (SUP.9) aus ASPICE 3.1 und *Problemlösungs-Prozess für SW* (§ 9) aus EN 62304 für die weiteren Analysen lösungsorientierter gewesen.

Die teilweise sehr geringe Abdeckung zwischen den agilen Praktiken und den Prozessen/Aktivitäten der beiden Normen beruht zusätzlich zu den bereits erwähnten Ursachen auch auf der sehr kritischen Beurteilung. Dabei wurden Anforderungen aus den BPs bzw. Aufgaben nur als erfüllt bewertet, wenn die Umsetzung ohne größere Anpassungen der Praktiken möglich ist.

5.3 Entwicklung eines agilen Scrum- Lebenszyklus für ausgewählte Prozesse/Aktivitäten

Für den entwickelten Vorschlag eines agilen Lebenszyklus (Abb. 4.1 und 4.2) für die ausgewählten Prozesse/Aktivitäten wurden Lösungen für die entstandenen Lücken gesucht, indem bestehende agile Artefakte aus Scrum und Kanban angepasst und zusätzliche nicht-agile Methoden vorgeschlagen wurden. Diese Artefakte, Methoden und die Beschreibung zur Erfüllung der Anforderungen aus den Normen sind in Tab. 9 - 11 zusammengefasst. Die im zweiten Lebenszyklus verwendeten Praktiken *Änderungsmanagementplan* und *Problemmanagementplan* befinden sich am Rande des Lebenszyklus, da sie am Anfang des Projektes erstellt werden und Änderungen/Probleme während des Projektes nach dessen Vorgaben behandelt werden müssen. Weitere Anforderungen, wie etwa die Herstellung von Traceability oder die Kategorisierung von Anforderungen, wurden durch das Tool JIRA Software (Atlassian Corp. Plc, Sydney, Australien) und allen darin enthaltenenen Funktionen gewährleistet. JIRA Software ist eine webbasierte Anwendung, die in der agilen Gemeinschaft häufig zum Verfolgen von Bugs und anderen Anforderungen verwendet wird [34].

Aufgrund der Erkenntnis, dass die Analyse von Anforderungen ein länger andauernder Prozess ist, der nicht in einem Sprint erledigt werden kann, wurde der Lösungsvorschlag in zwei, in Reihe geschaltete Lebenszyklen, aufgeteilt. Der erste Lebenszyklus beschäftigt sich mit der Analyse und Dokumentation von Anforderungen laut SWE.1 und § 5.2. Ist die erste Anforderungsanalyse abgeschlossen, kann mit der Entwicklung der Software und der Bewältigung von Problemen und Änderungen fortgefahren werden. Um in folgenden

Studien die restlichen Prozesse und Aktivitäten aufnehmen zu können, müsste analysiert werden, ob diese in die bereits bestehenden Lebenszyklen integrierbar sind oder Weitere entwickelt werden müssen.

5.4 Agile Softwareentwicklung und V-Modell im Vergleich

Grundsätzlich haben sowohl das V-Modell als auch Scrum Vorteile in der Anwendung. Einerseits punktet Scrum mit Flexibilität, kurzer Entwicklungsdauer und Kunden- bzw. Mitarbeiterzufriedenheit, und andererseits das V-Modell mit hoher Produktqualität aufgrund des reglementierten Testverfahrens in jeder Anforderungsebene.

Es gibt jedoch mehrere Gründe, warum viele Unternehmen der Umstieg zur agilen Softwareentwicklung abschreckt. Um Scrum in einem Unternehmen ohne agiler Vorgeschichte etablieren zu können, müssten zusätzliche Methoden, Tools (z.B. JIRA) und Rollen, wie *Scrum Master* oder *Agile Coach* in der Firmenstruktur eingeführt werden. Weiters werden in vielen Unternehmen aufgrund von fehlender Unterstützung durch das Management oder Mangel an Erfahrung im Bereich agiler Entwicklung Methoden wie Scrum oder Kanban außer Acht gelassen. Die Implementierung einer neuen Softwareentwicklungmethode ist naturgemäß auch mit hohen Umstellungskosten verbunden.

Die Ergebnisse dieser Arbeit ergaben, dass die Softwareanforderungsanalyse (SWE.1/§ 5.2) aus dem V-Modell mit agilen Möglichkeiten großteils umsetzbar wäre. Um die weiteren Aktivitäten/Prozesse aus dem untersten Bereich des V-Modells (Abb. 1.13) auf Abdeckungsmöglichkeit durch agile Artefakte zu überprüfen, müssten die restlichen Aktivitäten/Prozesse einer detailierten Analyse unterzogen werden.

6 Schlussfolgerung

Mit der Analyse in dieser Arbeit kann zusammengefasst werden, dass Anforderungen der untersuchten Prozesse bzw. Aktivitäten (Anforderungs-, Änderungs- und Problemmanagement) aus EN 62304 und Automotive SPICE 3.1 durch die Anwendung und Anpassung von agilen Praktiken aus Scrum und Kanban, und zusätzlichen Methoden aus der Softwareentwicklung umgesetzt werden können. Ein alleiniger Einsatz der Artefakte aus Scrum und Kanban zur Erfüllung der Normanforderungen wäre nicht möglich, was sich mit den Ergebnissen aus [33] und [35] deckt. Beispielsweise konnte eine Umsetzung der Traceability zwischen Anforderungen durch agile Artefakte nicht abgedeckt werden.

In der Praxis wäre eine Umstellung vom V-Modell auf das agile Rahmenwerk Scrum in den analysierten Prozessen/Aktivitäten möglich, hätte aber viele Änderungen in der Firmenstruktur, wie z.B. die Einführung von zusätzlichen Methoden, Tools oder Rollen, zur Folge.

Weitere Analysen zur Abdeckung der restlichen Prozesse wären notwendig, um einen vollständigen Lebenszyklus zu entwickeln und die Anwendbarkeit der agilen Methoden in der Automobil- und Medizintechnik zu prüfen.

Literatur

[1] Sutherland, J.: Agile Principles and Values, https://msdn.microsoft.com/de-de/library/dd997578(v=vs.120).aspx, Abgerufen am: 03.02.2018.

[2] Stacey, R.D.: Strategic Management and Organisational Dynamics - The Challenge of Complexity. London, Pearson Education (2002).

[3] Kugler Maag CIE: Kugler Maag - Agile in Automotive - State of Practice 2015 survey. Kornwestheim, Kugler Maag CIE (2015).

[4] VersionOne Inc.: 12th annual state of agile report. Atlanta, VersionOne Inc. (2018).

[5] Sliger, M.: Agile project management with Scrum. Dallas, Project Management Institute (2011).

[6] Rubin, K.S.: Essential Scrum - A Practical Guide to the Most Popular Agile Process. Amsterdam, Addison-Wesley Longman (2012).

[7] Kniberg, H.: One day in Kanban land. http://blog.crisp.se/2009/06/26/henrikkniberg/1246053060000 (2009), Abgerufen am: 14.03.2018.

[8] VDA QMC Working Group 13 / Automotive SIG: Automotive SPICE Process Assessment/Reference Model 3.1. VDA QMC Working Group 13 / Automotive SIG (2017).

[9] ISO/IEC 33020:2015, Information technology - Process assessment - Process measurement framework for assessment of process capability. (2015).

[10] BTC Embedded Systems AG: IBM Rational Rhapsody Reference Workflow Guide. Oldenburg, BTC Embedded Systems AG (2015).

[11] Masak, D.: IT-Alignment - IT-Architektur und Organisation. Berlin Heidelberg, Springer Science & Business Media (2006).

[12] Larman, C., Basili, V.R.: Iterative and Incremental Development - a brief history. Computer. 2003, 36:47-56.

[13] Schwaber, K., Beedle, M.: Agile Software Development with Scrum. London, Prentice Hall (2002).

[14] Beck, K.: Extreme Programming Explained - Embrace Change. Boston, Addison-Wesley Professional (1999).

[15] Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile software development methods - Review and Analysis, VTT publication 478, 2002.

[16] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith,J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: Manifesto for Agile Software Development, http://agilemanifesto.org (2001), Abgerufen am: 03.02.2018.

[17] Roock, S., Pieper, F.-U.: Agile Verträge - Vertragsgestaltung bei agiler Entwicklung für Projektverantwortliche. Heidelberg, dpunkt.verlag (2017).

[18] Woher kommt der Name Scrum?, http://www.lean-agility.de/2015/04/woher-kommt-der-name-scrum.html??? (2015), Abgerufen am: 08.08.2018.

[19] Kugler Maag CIE: Agile in Automotive - Scrum Pocket Guide. Kornwestheim, Kugler Maag CIE (2015).

[20] Mahalakshmi, M., Sundararajan, M.: Traditional SDLC Vs Scrum Methodology - A Comparative Study. Int. J. Emerging Technology and Advanced Engineering 2013, 6:192-196.

[21] Strube, D.: Scrum, Kanban und Co.: Starke Teams ersetzen starre Prozesse, iX Developer - Embedded Software 2014, 2:158-163.

[22] IX-Redaktion: iX Special 2017 – IT-Projektmanagement: Agil bessere Software entwickeln. Hannover, Heise Medien GmbH & Co. KG (2017).

[23] Davies, R., Sedley, L.: Agiles Coaching - Praxis-Handbuch für ScrumMaster, Teamleiter und Projektmanager in der agilen Software-Entwicklung. Heidelberg, MITP-Verlags GmbH & Co. KG (2010).

[24] Schwaber, K., Sutherland, J.: The Scrum Guide - The Definition Guide to Scrum:

The Rules of the Game, http://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf (2017), Abgerufen am: 10.02.2018.

[25] Dräther, R., Koschek, H., Sahling, C.: Scrum kurz & gut. Köln, O'Reilly Verlag GmbH & Co KG (2013).

[26] Anderson, D.J.: Kanban - Evolutionäres Change Management für IT-Organisationen. Heidelberg, dpunkt.verlag (2011).

[27] Leopold, K.: Kanban in der Praxis - Vom Teamfokus zur Wertschöpfung. München, Carl Hanser Verlag GmbH Co KG (2016).

[28] Leopold, K., Kaltenecker, S.: Kanban in der IT - Eine Kultur der kontinuierlichen Verbesserung schaffen. München, Carl Hanser Verlag GmbH Co KG (2018) , M.

[29] Boral, S.: Ace the PMI-ACP® exam - A Quick Reference Guide for the Busy Professional. New York, Apress Media LLC (2016).

[30] DIN ISO/IEC 15504:2011-07, Informationstechnik - Prozess-Assessment (Teil 1-6). (2011).

[31] SN EN 62304:2006-07, Medizingeräte-Software - Software-Lebenszyklus-Prozesse (2006).

[32] Scrum & Kanban & V-Modell, https://www.telemotive.de/de/softwareentwicklung/methodik/scrum-kanban-v-modell/ (2017), Abgerufen am: 21.10.2018.

[33] Kugler Maag CIE: Kugler Maag - Automotive SPICE (R) v3.1 - Pocket Guide - Extended VDA Scope. Kornwestheim, Kugler Maag CIE (2018).

[34] Bahr, I.: 6 der besten Softwarelösungen für das agile Projektmanagement. https://www.capterra.com.de/blog/258/7-top-agile-projektmanagement-tools (2018), Abgerufen am: 20.08.2018.

[35] Bless, M.: Scrum und IEC 62304: Medizinische Software mit agilen Methoden normkonform entwickeln. Berlin, epubli GmbH (2014).

7 Anhang

Vergleich zwischen § 5.1 (EN 62304) & MAN.3 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
5.1.1 Software-Entwicklungsplan	MAN.3: BP1, BP2 (Largely)	2
5.1.2 Aktualisierung des Software-Entwicklungsplans	MAN.3: BP4 (Fully)	3
5.1.3 Referenz im Software-Entwicklungsplan auf SYSTEM-Design & -Entwicklung	MAN.3 (Not achieved)	0
5.1.4 Planung von Normen, Methoden und Werkzeugen der SW-Entwicklung	MAN.3: BP6 (Partially)	1
5.1.5 Planung der SW-Integration & Integrationsprüfung	MAN.3 (Not achieved)	0
5.1.6 Planung der SW-Verifizierung	MAN.3 (Not achieved)	0
5.1.7 Planung der SW-Risikomanagements	MAN.3 (Not achieved)	0
5.1.8 Planung der Doku	MAN.3 (Not achieved)	0
5.1.9 Planung des SW-Konfigurationsmanagements	MAN.3 (Not achieved)	0
5.1.10 Zu kontrollierende unterstützende Komponenten	MAN.3 (Not achieved)	0
5.1.11 Kontrolle der Software-KONFIGURATIONSELEMENTE vor der Verifizierung	MAN.3 (Not achieved)	0

Vergleich zwischen § 5.2 (EN 62304) & SWE.1 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
5.2.1 Ableitung der Software-Anforderungen aus den SYSTEM-Anforderungen und Dokumentation	SWE.1: BP1 (Fully)	3
5.2.2 Inhalt der Software-Anforderungen	SWE.1: BP1, BP4 (Largely)	2
5.2.3 Einbeziehen von RISIKOKONTROLL-Maßnahmen in die Software-Anforderungen	SWE.1 (Not achieved)	0
5.2.4 Erneute EVALUATION der RISIKOANALYSE	SWE.1 (Not achieved)	0
5.2.5 Aktualisierung von SYSTEM-Anforderungen	SWE.1: BP7 (Fully)	3
5.2.6 VERIFIZIERUNG von Software-Anforderungen	SWE.1: BP3, BP5, BP6 (Fully)	3

Vergleich zwischen § 5.3 (EN 62304) & SWE.2 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
5.3.1 Umsetzung von Software-Anforderungen in einer ARCHITEKTUR	SWE.2: BP1, BP2 (Fully)	3
5.3.2 Entwicklung einer ARCHITEKTUR für die Schnittstellen zwischen SW-KOMPONENTEN	SWE.2: BP3, BP6, BP7 (Fully)	3
5.3.3 Spezifikation der Funktions- und Leistungsanforderungen für SOUP-Komponenten	SWE.2 (Not achieved)	0
5.3.4 Spezifikation der für die SOUP-Komponente erforderliche SYSTEM-Hardware und -Software	SWE.2 (Not achieved)	0
5.3.5 Festlegung der für die RISIKOBEHERRSCHUNG erforderliche Abgrenzung	SWE.2 (Not achieved)	0
5.3.6 VERIFIZIERUNG der SW-ARCHITEKTUR	SWE.2: BP6, BP7, BP8 (Largely)	2

Vergleich zwischen § 5.4 (EN 62304) & SWE.3 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
5.4.1 Feinaufteilung der Software-ARCHITEKTUR in SW-EINHEITEN	SWE.3: BP1 (Fully)	3
5.4.2 Entwicklung eines detaillierten Designs für jede SW-EINHEIT	SWE.3: BP1 (Fully)	3
5.4.3 Entwicklung eines detaillierten Designs für Schnittstellen	SWE.3: BP2 (Fully)	3
5.4.4 VERIFIZIERUNG des detaillierten Designs	SWE.3: BP4, BP5, BP6 (Fully)	3

Vergleich zwischen § 5.5 (EN 62304) & SWE.4 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
5.5.1 Implementierung jeder SW-EINHEIT	SWE.4 (Not achieved)	0
5.5.2 Festlegung eines VERIFZIERUNGSPROZESSES für SW-EINHEITEN	SWE.4: BP2 (Fully)	3
5.5.3 Akzeptanzkriterien für SW-EINHEITEN	SWE.4: BP2 (Fully)	3
5.5.4 Zusätzliche Akzeptanzkriterien für SW-EINHEITEN	SWE.4 (Not achieved)	0
5.5.5 VERIFIZIERUNG der SW-EINHEITEN	SWE.4: BP3 (Fully)	3

Vergleich zwischen § 5.6 (EN 62304) & SWE.5 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
5.6.1 Integration der SW-EINHEITEN	SWE.5: BP1 (Fully)	3
5.6.2 VERIFIZIERUNG der SW-Integration	SWE.5: BP4 (Largely)	2
5.6.3 Prüfung der integrierten SW	SWE.5: BP6 (Fully)	3
5.6.4 Inhalt der Integrationsprüfung	SWE.5: BP6 (Largely)	2
5.6.5 VERIFIZIERUNG der SW-EINHEITEN	SWE.5 (Not achieved)	0
5.6.6 Durchführung von REGRESSIONSPRÜFUNGEN	SWE.5: BP2 (Fully)	3
5.6.7 Inhalt von Aufzeichnungen über die Integrationsprüfung	SWE.5: BP9 (Largely)	2
5.6.8 Verwendung eines Problemlösungs-PROZESSES für SW	SWE.5 (Not achieved)	0

Vergleich zwischen § 5.7 (EN 62304) & SWE.6 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
5.7.1 Festlegung von Prüfungen für SW-Anforderungen	SWE.6: BP2 (Fully)	3
5.7.2 Verwendung eines Problemlösungs-PROZESSES für SW	SWE.6 (Not achieved)	0
5.7.3 Prüfungswiederholung nach Änderungen	SWE.6 (Not achieved)	0
5.7.4 VERIFIZIERUNG der SW-SYSTEM-Prüfungen	SWE.6: BP2 (Fully)	3
5.7.5 Inhalte der Aufzeichnungen der SW-SYSTEM-Prüfungen	SWE.6: BP7 (Largely)	2

Kein Vergleich zwischen § 5.8 (EN 62304) und ASPICE 3.1 Prozess		
Aufgabe aus EN 62304	Begründung	ÜG
5.8.1 Sicherstellen, dass die VERIFZIERUNG der SW vollständig ist	Keine vorgeschriebenen Prozesse/BPs zur Freigabe von SW im VDA Scope von Automotive SPICE. Außerhalb des VDA Scopes: eventuell SPL.2	0
5.8.2 Dokumentation bekannter restlicher ANOMALIEN		
5.8.3 Berwertung bekannter restlicher ANOMALIEN		
5.8.4 Dokumenation freigegebener VERSIONEN		
5.8.5 Dokumenation, wie freigegebene SW erzeugt wurde		
5.8.6 Sicherstellen, dass AKTIVITÄTEN und AUFGABEN abgeschlossen sind		
5.8.7 Archivierung der SW		
5.8.8 Sicherstellen der Wiederholbarkeit der SW-Freigabe		

Vergleich zwischen § 8.1 (EN 62304) & SUP.8 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
8.1.1 Festlegung von Mitteln zur Identifizierung von KONFIGURATIONSELEMENTEN	SUP.8: BP2 (Fully)	3
8.1.2 Identifizierung von SOUP	SUP.8 (Not achieved)	0
8.1.3 Identifizierung der Dokumenation der SYSTEM-Konfiguration	SUP.8: BP2 (Fully)	3

Vergleich zwischen § 8.2 (EN 62304) & SUP.10 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
8.2.1 Genehmigung von ÄNDERUNGSANFORDERUNGEN	SUP.10: BP4, BP5 (Fully)	3
8.2.2 Implementierung von Änderungen	SUP.10: BP4, BP5 (Fully)	3
8.2.3 VERIFIZIERUNG von Änderungen	SUP.10: BP6 (Fully)	3
8.2.4 Bereitstellung von Mitteln für die RÜCKVERFOLGBARKEIT von Änderungen	SUP.10: BP7 (Partially)	1

Vergleich zwischen § 8.3 (EN 62304) & SUP.8 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
8.3 Aufzeichungen über den Status der Konfiguration	SUP.8: BP6, BP8, BP9 (Fully)	3

Vergleich zwischen § 9.1 (EN 62304) & SUP.9 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
9.1 Erstellen von PROBLEMBERICHTEN	SUP.9: BP2, BP4 (Fully)	3

Vergleich zwischen § 9.2 (EN 62304) & SUP.9 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
9.2 Untersuchung des Problems	SUP.9: BP4 (Fully)	3

Vergleich zwischen § 9.3 (EN 62304) & SUP.9 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
9.3 Unterrichtung beteiligter Stellen	SUP.9 (Not achieved)	0

Vergleich zwischen § 9.4 (EN 62304) & SUP.10 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
9.4 Anwendung des Änderungskontroll-PROZESSES	SUP.10 Bezug auf 8.2. 80% (Largely)	2

Vergleich zwischen § 9.5 (EN 62304) & SUP.9 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
9.5 Aufbewahrung von Aufzeichnungen	SUP.9: BP2, BP3 (Fully)	3

Vergleich zwischen § 9.6 (EN 62304) & SUP.9 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
9.6 Analyse von Problemen hinsichtlich Trends	SUP.9: BP9 (Fully)	3

Vergleich zwischen § 9.7 (EN 62304) & SUP.9 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
9.7 VERIFIZIERUNG der Lösung von SW-Problemen	SUP.9: BP7 (Fully)	3

Vergleich zwischen § 9.8 (EN 62304) & SUP.9 (ASPICE 3.1)		
Aufgabe aus EN 62304	Begründung	ÜG
9.8 Inhalt von Prüfungsdokumentation	SUP.9 (Not achieved)	0