Sonderdruck aus OBJEKTspektrum

Requirements Engineering in agilen Projekten

Verständnis durch story-spezifische Artefakte

Sowohl für agiles Entwickeln als auch agiles Testen gibt es etablierte Techniken und ein grundlegendes gemeinsames Verständnis in den jeweiligen Communities. Im Requirements Engineering hingegen stellen sich Projekte Fragen wie: Was passiert im Zuge einer agilen Transformation mit den Artefakten des RE (wie formalen Dokumenten oder grafischen Modellen) und der Rolle des Requirements Engineers? Bedeutet „agiles RE" tatsächlich, Anforderungen nur noch in Form von User-Stories zu erfassen? Wie passt die Rolle eines Product Owner ins Bild? Und wie drückt sich ein wirklich agiles RE-Vorgehen im Projektalltag aus, konkret beispielsweise in einem Scrum-Sprint-Planning?

Dass eine allgemein akzeptierte Definition von „agilem RE" noch nicht anzutreffen ist, ist für sich alleine schon eine spannende Beobachtung. Sie drückt sich aus in Publikationen der letzten Monate wie „Agiles Requirements Engineering – ein Oxymoron?" [Hellw] oder „Modellbasiertes Requirements Engineering: Geht das auch agil?" [Kra18]. Der IREB-Lehrplan zum „RE@Agile Primer" [IREB] bietet übrigens derzeit ebenfalls keine geschlossene Begriffsklärung, sondern „[…] bevorzugt den Begriff RE@Agile gegenüber ‚Agiles Requirements Engineering', um zu verdeutlichen, dass das RE prozessunabhängig ist". Und in Projekten beobachte ich unter anderem:

- Typische RE-Artefakte tauchen in agilen Teams kaum noch auf. Kein Wunder: Dokumentation gilt häufig als „No Go", egal ob es sich um textuelle Spezifikationen oder Modelle handelt. Stories und direkte Kommunikation sollen genügen.
- Die Rolle eines Requirements Engineers ist in agilen Teams ebenfalls so gut wie nie anzutreffen. Spannend daran: Viele seiner Aufgaben sind zur Rolle des Product Owner (PO) „gewandert", ohne dass diese Rolle dies immer explizit so wahrnimmt, geschweige denn sich fragt, welche Hilfsmittel zu diesen Aufgaben denn eventuell einen Blick lohnen würden. (Dass es sich hierbei um zwei verschiedene Rollen handelt, werde ich später erörtern.)
- Und Stories im Projektalltag sind sehr oft verbesserungswürdig, da zu groß, inhaltlich schon im Lösungsraum angesiedelt, nicht mit testbaren Akzeptanzkriterien ausgestattet usw.

Auf einer RE-Konferenz 2018 mit Schwerpunkt Agilität in Berlin [ModRE] führte ich in meinem Vortrag ein kleines Experiment durch: Ich befragte das anwesende Fachpublikum – ca. 80 RE-Experten – nach ihrer Ansicht zu „agilem RE" und bot drei Antworten zur Auswahl:

- „Mir ist klar, was agiles RE bedeutet, und ich könnte es jederzeit erklären." (2 Hände erhoben sich für diese Option)
- „Sowas gibt's eigentlich gar nicht." (3 Hände)
- „Mir ist's *nicht* klar, und ich hätte gerne mal eine Antwort darauf." (Alle anderen!)

Repräsentativ oder nicht, als Signal und Stimmungsbild spricht das Ergebnis allemal eine deutliche Sprache. Und als ich die Befragung mit der – bewusst provokanten – Behauptung abschloss, dass agiles Entwickeln seit über 10 Jahren und agiles Testen seit etwa 5 Jahren verstanden und etabliert, das agile Requirements Engineering im Vergleich dazu aber noch deutlich im Rückstand sei, war die Reaktion des Publikums umfassende Zustimmung und nicht die kleinste Spur von Protest.

Ich werde in diesem Beitrag also eine Antwort auf die Frage entwickeln, was „agiles RE" ist oder sein kann, wie es sich vom klassischen RE unterscheidet und wie es sich im agilen Projektalltag konkret darstellen kann. Insbesondere werde ich zeigen, dass es sehr wohl möglich ist, agiles Vorgehen und „klassische" RE-Artefakte praxistauglich und nutzbringend zusammenzuführen.

Zur Vorbereitung benötige ich vier „Thesen" als Puzzleteile, die sich im Anschluss zu einem funktionierenden agilen RE-Vorgehen zusammensetzen lassen werden. Beginnen wir mit einem weit verbreiteten Missverständnis über User-Stories.

These 1: User-Stories sind keine Requirements

Sollten Sie jetzt völlig überrascht sein: Ein eigener Artikel mit fast gleich lautender Überschrift [NoReq] begründet diese Position ausführlich, sodass ich mich hier auf eine Zusammenfassung beschränken kann.

In Kurzform: Eine User-Story ist ein *Planungsartefakt* und trägt als Titel einen leicht verständlichen Ausdruck eines Kundenwunsches (ausdrücklich im *Problemraum* und NICHT im Lösungsraum angesiedelt), ergänzt um Akzeptanzkriterien. Eine Story enthält aber nicht alle Details, die zum Entwickeln und Testen der Lösung nötig sind. Ron Jeffries [CCC] hat in seinem CCC-Prinzip darauf hingewiesen, dass Stories im Gegensatz zu dokumentierten Requirements etwas „Soziales" haben, da sie vor allem die Zusammenarbeit der Projektbeteiligten fördern sollen: Von den drei „C"s (Card – Conversation – Confirmation) bedeutet das mittlere C, dass an Stelle umfangreicher Dokumentation die direkte Kommunikation zu bevorzugen ist.

Das bedeutet: Eine User-Story ist nur eine *Überschrift* für ein Requirement, keinesfalls aber eine Requirement-Spezifikation. Sie ist, anders gesagt, ein ausführlicher und bewusst sprechend gestalteter Name für eine Anforderung.

Um die Anforderungsdetails wiederum zu klären, gibt es zwei Möglichkeiten: *direktes Gespräch* oder *Dokumentation*. Agile Teams bevorzugen zurecht ersteres, aber es gibt definitiv Situationen, in denen das gesprochene Wort allein entweder nicht genügt oder etwas mehr Dokumentation dem Team zweifellos bei der Arbeit helfen würde. Diesen Gedanken werden wir gleich noch einmal aufgreifen.

Halten wir fest: User-Stories *benennen* Anforderungen, aber sie *beschreiben* sie nicht.

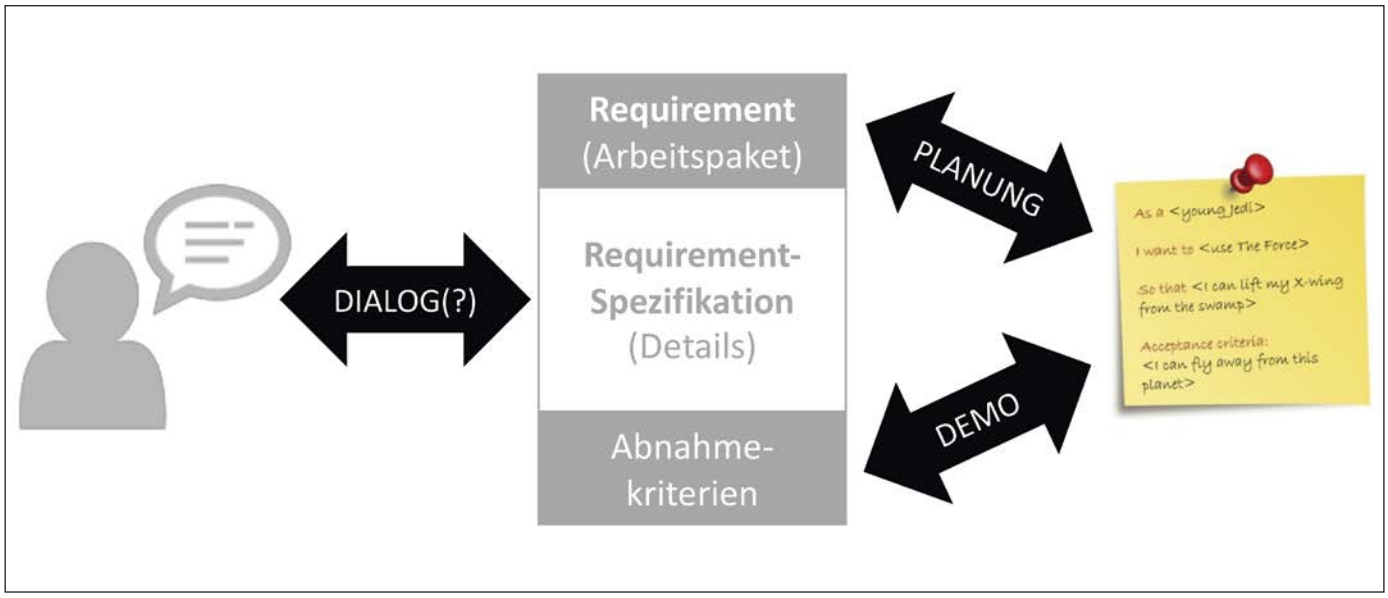

Abb. 1: Story = Arbeitspaket + Akzeptanzkriterien

These 2: Agiles Dokumentieren bedeutet: Nur dokumentieren, was einen Wert oder Nutzen hat

Das agile Manifest [Manif], genauer: sein zweites „Wertepaar", ist vermutlich einer der Hauptgründe dafür, dass sich agiles Requirements Engineering in vielen Projekten nur schwer entwickeln konnte. Die Aussage „Funktionierende Software ist ein höherer Wert als umfangreiche Dokumentation" ließ sich von Anfang an leicht missverstehen als „im Agilen wird nicht mehr dokumentiert". Tatsächlich ist die korrekte Bedeutung eine andere: Im Agilen soll nichts nur um des Dokumentierens willen dokumentiert werden. Oder noch einmal anders formuliert: *Es wird im Agilen nur dokumentiert, was für irgendjemanden einen Wert oder Nutzen hat!*
Wie aber entsteht der „Wert oder Nutzen" eines Dokuments? Entweder

- das Dokument ist NOTWENDIG (für regulatorische Zwecke wie Audits, für Wartung und Pflege, als Teil der Produktübergabe usw.) oder
- das Dokument ist HILFREICH (für das Team bei der Erledigung seiner Arbeit).

Ist beides nicht gegeben, besteht kein Grund, auch nur ein Quäntchen Aufwand in die Erstellung eines Dokuments fließen zu lassen. Nehmen wir diese beiden möglichen Motivationen für Dokumentation also als zweite These mit.

These 3: Je nach Anforderung helfen unterschiedliche Beschreibungsformate unterschiedlich gut

Abbildung 2 zeigt einige typische Formate, mit deren Hilfe Anforderungen formuliert und dokumentiert werden können. Diese muss ich hier nicht im Detail erklären, stattdessen möchte ich als Ergänzung eine weitere subjektive Beobachtung einfließen lassen:

- Die Form der „gesprochenen Auskunft" wird in agilen Teams bevorzugt (zur Begründung siehe These 2).
- Die Form der „natürlichsprachlichen Dokumentation" war und ist ein natürlicher „Default" in zahlreichen nicht-agilen Projekten.
- Alle übrigen Formate werden sowohl von vielen agilen als auch nicht-agilen Projektteams leider ignoriert.

Letzteres hat natürlich Gründe – vom Skillset im jeweiligen Team über den befürchteten Mehraufwand bis hin zu sperrigen Werkzeugen. Trotzdem habe ich mehrfach in Projekten die Erfahrung machen dürfen, dass es zu gewissen Anfor-

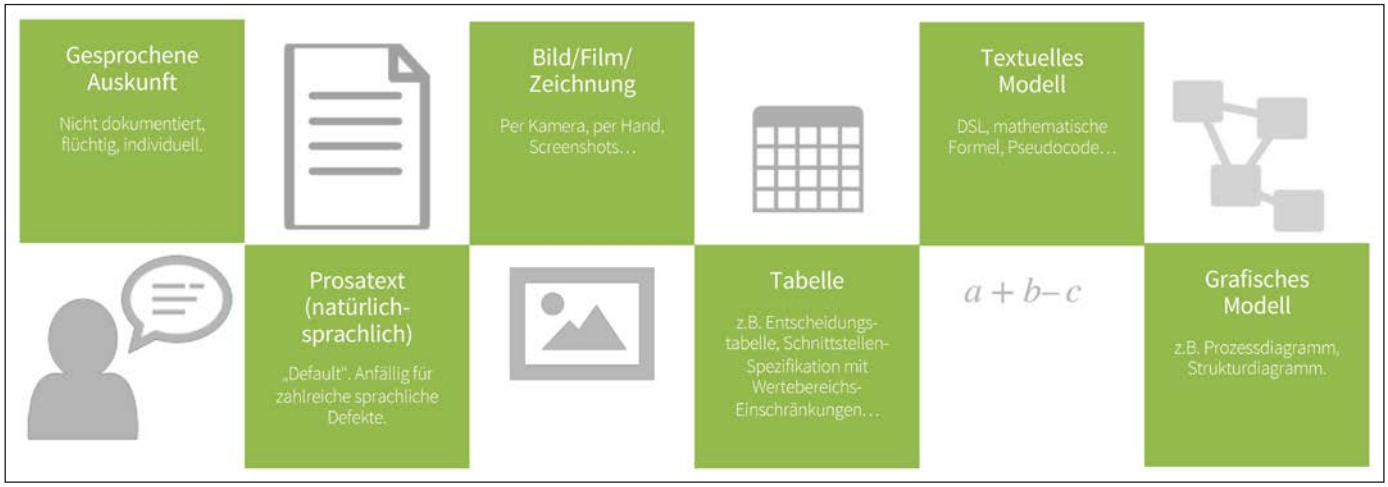

Abb. 2: Beschreibungsformate für Requirements

derungsgegenständen Notationsformen gibt, die „wie für sie geschaffen“ und natürlicher Sprache mühelos überlegen sind :

- einen *Prozess* stelle ich als *Diagramm* dar (und beschreibe ihn nicht textuell);
- komplexe *Wenn-dann-Beziehungen* können am besten als *Entscheidungstabelle* beschrieben (und getestet!) werden, während man sich das textuelle Pendant schon nach wenigen Minuten nicht mehr antun mag;
- *Berechnungen* lassen sich klar und unmissverständlich als mathematische *Formel* spezifizieren; und
- Satzschablonen unterstützen bei vielen nicht-funktionalen Anforderungen.

Für jedes Töpfchen gibt es also im Fundus „klassischer“ RE-Techniken oftmals ein passendes Deckelchen, das die Anforderungsqualität verbessern kann.
Und ganz wichtig: Wenn die gesprochene Auskunft dem Team genügt, um alle benötigten Details zu einer konkreten User-Story zu bekommen – belassen Sie es dabei! Der Appell geht also dahin, dies jeweils pro Story im Einzelfall zu entscheiden.
Und damit zur letzten vorbereitenden These:

These 4: Ziel des agilen RE muss sein, zweimal Verständnis im Team zu erzeugen

Der Satz ist sicherlich erklärungsbedürftig (vor allem das „zweimal“). Vergessen wir einmal RE-Definitionen wie „Anforderungen erheben, dokumentieren, managen usw.“ und fragen uns stattdessen: Wozu das Ganze? Die Antwort lautet: Das Projektteam muss verstehen, *was* der Kunde/Nutzer benötigt und *warum* (eben den sogenannten Problemraum). Gutes RE hat also das *Verständnis* des Teams im Blick und wird damit zum zentralen Erfolgsfaktor.
Im Agilen schälen sich schnell zwei Themen heraus, für die ein Team Verständnis entwickeln sollte. Diese ergeben sich unmittelbar aus der agilen Grundidee, ein Projekt nicht vor der Entwicklung detailliert auszuspezifizieren und durchzuplanen, sondern in Kenntnis des Ziels(!) jeweils nur den nächsten Schritt (=die nächste Iteration, beispielsweise einen Sprint) zu planen, das Ergebnis zu demonstrieren und das Feedback beziehungsweise Änderungen in den Anforderungen im nächsten Schritt zu berücksichtigen. Damit steht die vierte These im Grunde schon da: Ein agiles Team braucht Verständnis über

1. das Ziel (die Produktvision, das Big Picture, das fertige Produkt) und
2. die nächste Iteration (konkret: Story-Details).

Alle Maßnahmen, die dieses Verständnis mehren, sind wertvoll und deshalb willkommen.

Wenn Sie allen vier Thesen bis hierher folgen (und vielleicht sogar zustimmen) konnten, sind wir nun soweit, sie zu einem agilem RE-Vorgehen zusammenzusetzen.

Das Endprodukt verstehen

In der Regel trifft sich ein Projektteam vor der ersten Iteration, um in Workshops, Kickoff-Meetings oder dergleichen auf ein gemeinsames Ziel eingeschworen zu werden. Es ist wichtig, dass alle Projektmitglieder ein möglichst einheitliches Bild davon im Kopf haben, wie das Endprodukt aussehen soll und warum. Die Disziplin der Business-Analysis, die RE-Aktivitäten zuweilen vorangeht, kann hierzu hilfreichen Input liefern – etwa in Form von

- Use-Cases und Use-Case-Abläufen/ “happy paths“ (nicht alle, aber die wichtigsten),
- Geschäftszielen (dito),
- Produktrisiken (dito),
- Kontext des Produkts und seiner Nutzung,
- geplanten Usern, etwa als Personas (dito),
- Proto- und Prätotypen, etwa entlang zentraler Use-Case-Abläufe (an das Pareto-Prinzip denken!),
- bis hin zu einem MVP (Minimum Viable Product).

All dies kann einem Team helfen, eine gemeinsame Produktvision zu entwickeln, und dann ist es eine gute Idee, die „wichtigsten“/essenziellen Informationen im Teamraum zu versammeln, etwa als Product Canvas [Canv] oder Produkt-Poster [QuHe].
Das Problem, dem Team eine Produktvision zu vermitteln, ist also weitgehend verstanden und gelöst. Wenden wir uns deshalb dem zweiten Thema zu, das ein Team einheitlich und gut verstehen muss.

Die Stories der nächsten Iteration verstehen

Im agilen Testen gibt es seit mehreren Jahren ein einfaches und elegantes Hilfsmittel zur Beschreibung der Teststrategie in agilen Teams: die sogenannten „Agilen Testquadranten“ nach Brian Marick [Quadr]. Sie listen verschiedene Testarten auf und sortieren sie entlang von zwei Dimensionen ein, die vier Quadranten aufspannen:

- fachliche vs. technische Tests,
- Tests aus Team- vs. Tests aus Nutzer-Perspektive.

Die Frage „Wie wird bei uns im Team getestet?“ erhält damit ein schönes Überblicks-Bild. Aber es geht noch mehr: Im Sprint Planning können zu jeder Story die benötigten Testaktivitäten vom Team aus den Quadranten ausgewählt und geschätzt werden, was nicht nur den Testaufwand transparent werden lässt, sondern außerdem die Schätzqualität verbessert.
Unsere ursprüngliche Idee bestand nun darin, ein hierzu analoges Hilfsmittel für RE-Aufgaben im Team zu schaffen. Nach mehreren Anläufen lautete die ernüchternde Erkenntnis: Eine 1:1-Übertragung der Testquadranten scheint nicht ohne Weiteres sinnvoll. Hauptgrund ist, dass die Testquadranten auch auf Artefakte des sogenannten *Lösungsraums* zugreifen, um daraus Tests abzuleiten – einer Klasse von Arbeitsergebnissen, die uns im agilen RE nicht zur Verfügung steht.
Also fingen wir von vorne an und stellten einer hypothetischen neuen Story, mit der sich ein Projektteam beschäftigen soll, die Fragen aus These 2: Gibt es Artefakte, die zusammen mit der Story entstehen *müssen*, oder andere, die dem Team beim Verständnis der Story *helfen* würden? Oder ist beides nicht der Fall, und direkte Kommunikation, etwa mit dem PO, genügt für diese Story?
Der erste Teil – *benötigte* Artefakte – fällt ebenfalls unter „bereits gelöstes Problem“: Diese sollten Bestandteil der *Definition of Done* sein. Die „DoD“ beschreibt, wann eine Story fertig, das heißt „fertig entwickelt + fertig getestet + fertig dokumentiert“ ist! Haken dran.
Wie aber findet man Artefakte, die *hilfreich* für das Team sein könnten? Idee: Man versammelt verschiedene Artefakttypen in geeigneter Form und stellt sie dem Team zur Auswahl bereit, ähnlich den agilen Testquadranten. Das Ergebnis war ein Poster namens „Helpful Artifact Map“, und sein Aufbau, der sich an den Notationsformen aus unserer 3. These orientieren kann, ist in **Abbildung 3** zu sehen.
Dieses Artefaktposter sieht natürlich in jedem Team unterschiedlich aus. Operativ lässt es sich wie die Marickschen Testquadranten nutzen: Zu einer entstehenden Story fragen sich PO und Team, was bei der Umsetzung helfen könnte – ein Pro-

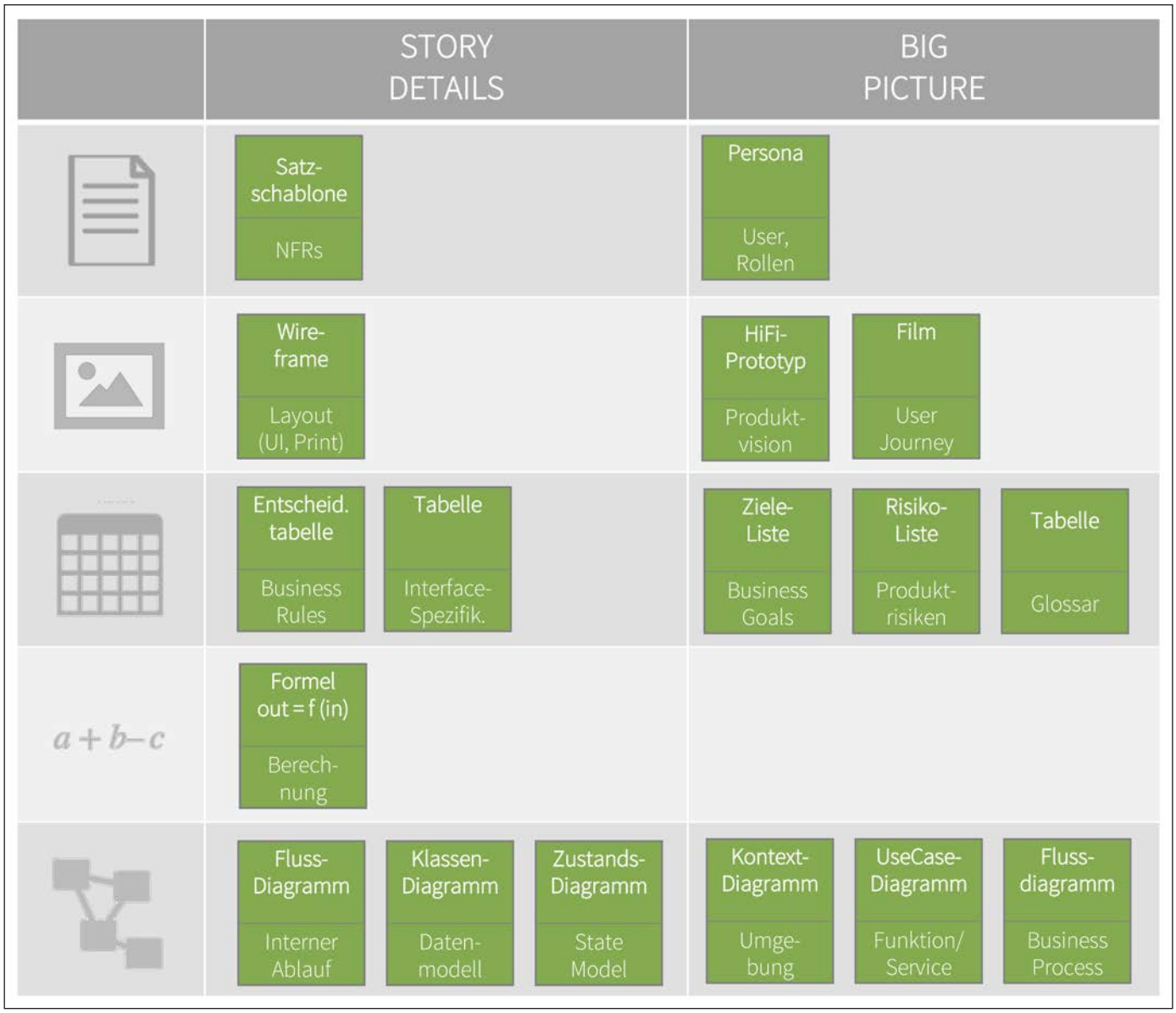

Abb. 3: Helpful Artifact Map

zessdiagramm, eine Entscheidungstabelle, eine Skizze. Wird etwas gefunden, wird es im Zuge des sogenannten Backlog Refinements/Groomings als Vorbereitung zum Sprint Planning erstellt und dem Arbeitspaket/der Story als Detail oder Referenz hinzugefügt.

Um die Auswahl zu erleichtern, empfehlen wir „Artefaktkarten" mit folgendem Aufbau: Zu jedem Artefakttyp (z. B. Entscheidungstabelle)

- wird ein *Beispiel* präsentiert (für die, die mit dem Artefakttyp noch nicht vertraut sind),
- werden *typische Situationen* versammelt, in denen das Artefakt nützlich sein kann,
- und *Testmethoden* genannt, die beim Test des entstandenen Artefakts infrage kommen können.

In **Abbildung 4** ist eine solche Artefaktkarte zu sehen.

An dieser Stelle werde ich normalerweise gefragt, *wer* denn diese Artefakte/Dokumente zu neuen User-Stories erstellt. Wenn wir uns beispielsweise die Rollen in Scrum anschauen, kommen ja nur der PO oder das Team selbst in Frage. Das Rätsel löst sich, wenn man sich daran erinnert, dass es in einem agilen Team nicht darum geht, dass „jedes Teammitglied alles kann", sondern darum, dass das Team als Ganzes alle Skills benötigt, die zur Bewältigung des Projekts benötigt werden. Außerdem ist es wichtig, zwischen Aufgaben und Rolle zu trennen und sich vor Augen zu halten, dass die PO-Rolle deutlich anders konzipiert ist als die eines Requirements Engineers! (Ein PO fällt etwa „major business decisions", um nur einen Unterschied zu nennen. Oder um Ken Schwaber zu zitieren: „In no way did I envision the Product Owner becoming a business analyst that was responsible for requirements engineering!" [Schwa])

Wenn im Refinement also erkannt wird, dass zum Beispiel ein Prozessdiagramm hilfreich wäre, gibt es zwei Möglichkeiten: Entweder der PO hat die nötigen Kenntnisse und erstellt das Diagramm selbst, oder er bittet jemanden aus dem Team, ihn dabei zu unterstützen. Im Umkehrschluss heißt das: Teammitglieder mit RE-Wissen sind eine nützliche Verstärkung für jedes agile Team, und kein ausgebildeter Requirements Engineer braucht Jobängste zu haben, wenn er für Veränderungen und Dazulernen offen ist. (Insbesondere muss ein PO also nicht zwingend ein RE-Profi sein. Es wäre für einen PO aber auch kein Schaden, über RE-Möglichkeiten Bescheid zu wissen!

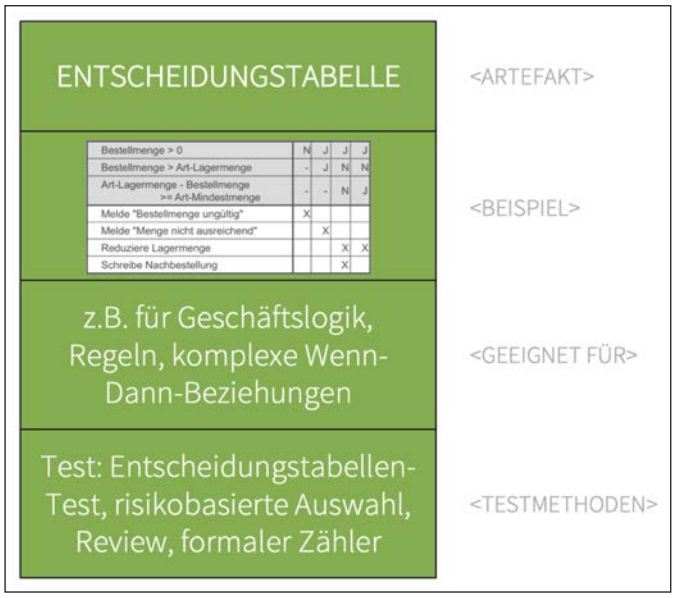

Abb. 4: Artefaktkarte

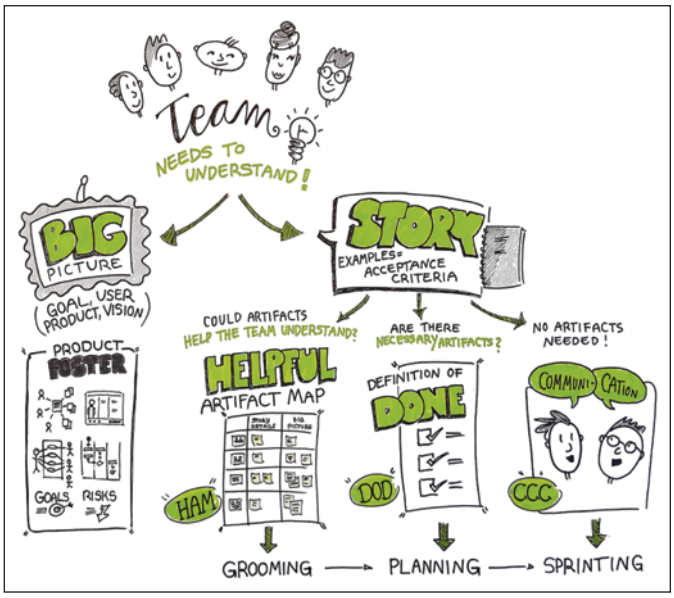

Abb. 5: Das Vorgehen im Überblick

Meine Empfehlung: Sie etablieren ein Pairing zwischen PO und RE-Kenner bei der Erstellung neuer User-Stories.)

Das Vorgehen im Überblick

Abbildung 5 zeigt das beschriebene Vorgehen als kompaktes Handout und versammelt alle genannten Hilfsmittel. Die meisten davon (Product Poster/Canvas, DoD, CCC-Prinzip) sind in vielen agilen Teams bekannt und bewährt. Neu hinzu kommt die „Helpful Artifact Map" mit ihren Artefaktkarten sowie der gesamte Entscheidungsbaum, der sich aus unseren vorbereitenden Thesen ableitet und die gezeigten Hilfsmittel ineinandergreifen lässt. (Eine „Tonspur" zur Abbildung erhalten Sie in [QuHe].)
Was erreichen wir mit diesem Vorgehen?

- Die üblichen „Slogans", die zur Beschreibung eines agilen RE herhalten müssen – „just enough documentation", „just-in-time Requirements", „so viel Dokumentation wie nötig und so wenig wie möglich" – sind allesamt erfüllt. Dies liegt an der Kernidee, *pro Story* zu entscheiden, ob eine dokumentierte Spezifikation notwendig oder hilfreich ist.
- Der Ansatz ist vollständig zum agilen Mindset kompatibel, da er mit leichtgewichtigen Hilfsmitteln arbeitet (Artefaktposter und -karten) und sich in jeder Retrospektive kritisch hinterfragen, erweitern und verbessern lässt.
- Er realisiert den agilen Grundgedanken, nicht zuerst eine umfangreiche und detaillierte Feinspezifikation zu erstellen und diese dann in Arbeitspakete zu zerlegen, sondern umgekehrt zu jedem Arbeitspaket (lies: jeder Story) lediglich die dazu benötigte oder hilfreiche Spezifikation vorzusehen.
- Und der Ansatz beseitigt die oft geäußerte Sorge, dass im Agilen benötigte formale Dokumentation keinen Platz mehr findet.

Übrigens: Auf der eingangs genannten RE-Konferenz [ModRE] wurde in einem Vortrag der Satz *„Requirements Engineering is dead – long live Requirements Communication!"* geprägt. Ich sehe darin keinen Widerspruch zum hier beschriebenen Vorgehen, sofern wir uns einig werden, was „Kommunikation" bedeutet: Ist damit ausschließlich verbale direkte Kommunikation gemeint? Oder allgemeiner der effektive und effiziente Informationsfluss von Sender zu Empfänger? Ich ziehe Letzteres vor, und genau dieses Bild – die Auswahl des hilfreichsten Formats zu einer gegebenen Story – finden Sie in unserem Ansatz wieder.
Letztlich drückt sich auch darin Agilität aus: Entscheidend ist, was dem Team hilft, egal, wo das Hilfsmittel herkommt – selbst wenn es sich zum Beispiel um ein Kontextdiagramm handelt, das der Strukturierten Analyse nach Tom deMarco aus den 70er Jahren entlehnt ist!

Als Dreingabe: Verbesserte Testbarkeit

Die Kirsche auf dem Kuchen stellt unseres Erachtens die mit dem Vorgehen geschaffene Brücke zwischen RE und Test dar. In [Bra16] finden Sie eine Unterscheidung zwischen fachlicher und technischer Testbarkeit; letztere ist vor allem ein Architekturthema, aber erstere beschäftigt sich mit der Frage, wie Anforderungen beschaffen sein müssen, damit der Aufwand für das Testdesign möglichst gering ausfällt. Die

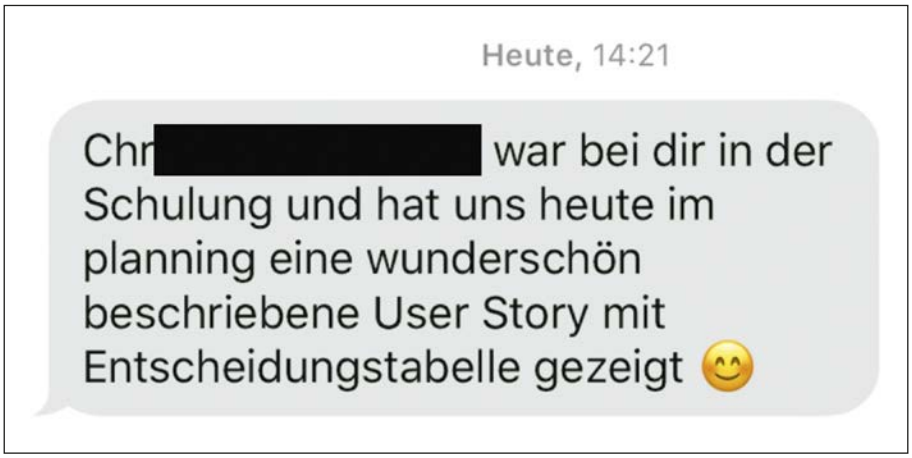

Abb. 6: It works!

Wir nennen ein Requirements Engineering (RE) „agil“, wenn es all diesen Prinzipien folgt:

- *Persönlichen Austausch bevorzugen:* Wann immer möglich und ausreichend, kommunizieren Endanwender, Produktverantwortlicher und Teammitglieder direkt miteinander. Dokumentiert wird nur, was für einen Stakeholder oder das Team nachweislich einen Wert/Nutzen hat.
- *Nur hilfreiche oder notwendige Spezifikationen erstellen:* Es werden RE-Artefakte ausgewählt und erstellt, die entweder HILFREICH für ein gemeinsames Team-Verständnis von (a) Endprodukt und (b) Arbeitspaketen der nächsten Iteration sind – oder die NOTWENDIG sind, um die Dokumentationsvorgaben von Stakeholdern zu erfüllen.
- *Details iterativ spezifizieren:* RE-Artefakte werden PRO ITERATION erstellt, das heißt, zu jedem Arbeitspaket (etwa einer Story) werden nur die benötigten oder hilfreichen Artefakte erstellt, die für die nächste Iteration benötigt werden („just-in-time requirements/just enough documentation“) – anstatt umgekehrt eine umfassende Spezifikation in Arbeitspakete zu zerlegen.
- *Aus Kundensicht priorisieren:* Anforderungen/Arbeitspakete werden gemäß Geschäftswert und Kundennutzen („business value/customer needs“) priorisiert.
- *Den Test unterstützen:* Die RE-Aktivitäten unterstützen die Erreichung von „fast feedback“ und testgetriebener Entwicklung (z. B. durch Akzeptanzkriterien, die mindestens demonstrierbare Beispiele oder selbst schon Testfälle sind, und durch Spezifikationen mit geeigneter Notation).
- *Kontinuierlich verbessern:* Das RE-Vorgehen wird regelmäßig hinterfragt und angepasst (z. B. in Retrospektiven).

Verantwortlich für die Auswahl, Erstellung und Pflege von Artefakten sowie für das RE-Vorgehen sind Produktverantwortlicher (z. B. PO) und Team.

Kasten 1: Agiles Requirements Engineering (Arbeitsstand)

beiden wichtigsten Unterstützungsfaktoren für fachliche Testbarkeit sind demnach

- *Akzeptanzkriterien*, die als konkrete Beispiele formuliert sind („Specification by Example“, [Adz11]),
- der gewählte *Formalismus* zur Anforderungsbeschreibung (siehe These 3 im vorliegenden Artikel).

Beide sind in unserem Ansatz vollständig integriert, und wir gehen sogar noch einen Schritt weiter: Die vorgeschlagenen Artefaktkarten (**siehe Abbildung 4**) listen explizit passende *Testentwurfsmethoden* zum gegebenen Artefakttyp auf – definitiv eine hilfreiche und „handfeste“ Verbindung zwischen RE und Test.

Fazit und Ausblick

Dass der Ansatz funktioniert, bestätigen Projektbeispiele. So bekam ich beispielsweise am Tag nach einem Workshop zu diesem Thema eine Nachricht von einem Scrum Master, die in **Abbildung 6** zu sehen ist – sie zeigt, dass der RE-Methodenbaukasten Lösungen bereitstellt, die hilfreich und klein genug für einen agilen Sprint sind. Und in einem anderen großen agilen Transformationsprojekt durfte ich im wahrsten Sinne des Wortes im Vorbeigehen an einem Daily Stand-up-Meeting den Satz eines Teammitglieds an den PO aufschnappen: „Das Diagramm, das du zu der Story erstellt hattest, hat mir gestern sehr geholfen.“

Um abschließend den Bogen zum Anfang des Beitrags zurückzuschlagen: In **Kasten 1** finden Sie unseren aktuellen Arbeitsstand zu einem Vorschlag, wie „agiles RE“ definiert werden könnte. Feedback, Fragen oder Anregungen hierzu sind herzlich willkommen. ||

Literatur & Links

[Adz11] G. Adzic, Specification By Example, Manning Publishing, 2011

[Bra16] C. Brandes, S. Okujava, J. Baier, Architektur und Testbarkeit: Eine Checkliste (nicht nur) für Softwarearchitekten, in: OBJEKTspektrum, 01&02/2016

[Canv] R. Pichler, The Product Canvas, 16.7.2012, siehe: **https://www.romanpichler.com/blog/the-product-canvas/**

[CCC] R. Jeffries' CCC-Prinzip: Essential XP: Card, Conversation, Confirmation, 30.8.2001, siehe: **http://xprogramming.com/articles/expcardconversationconfirmation/**

[Hellw] P. Hellwig, L. Grünz, Agiles Requirements Engineering – ein Oxymoron?, in: OBJEKTspektrum, Online Themenspecial, 2018, siehe: **https://www.sigs-datacom.de/ots/2018/re/4-agiles-requirements-engineering-ein-oxymoron.html**

[IREB] „RE@Agile Primer“ und „RE@Agile Glossar“, Internation Requirements Engineering Board, siehe: **https://www.ireb.org/de/downloads/**

[Kra18] A. Krallmann, Modellbasiertes Requirements Engineering: Geht das auch agil?, in: OBJEKTspektrum, 04/2018

[Manif] Agiles Manifest, siehe: **http://agilemanifesto.org/iso/de/manifesto.html**

[ModRE] ModernRE-Konferenz, siehe: **https://www.modern-re.de**

[NoReq] User Stories are not requirements, 23.5.2016, siehe: **http://www.scrumexpert.com/knowledge/user-stories-are-not-requirements/**

[Quadr] L. Crispin, Using the Agile Testing Quadrants, 8.11. 2011, siehe: **https://lisacrispin.com/2011/11/08/using-the-agile-testing-quadrants/**

[QuHe] QualityHeroes-Podcast Folge 9, 19.12.2018, siehe: **https://www.qualityminds.de/content/qualityheroes-podcast-folge-9**

[Schwa] K. Schwaber, Product Owners not proxies, 31.1.2011, siehe: **https://kenschwaber.wordpress.com/2011/01/31/product-owners-not-proxies**

Der Autor

Dr. Christian Brandes

(christian.brandes@qualityminds.de) arbeitet als Team Lead Requirements Engineering für die QualityMinds GmbH. Der promovierte Mathematiker und langjährige Testspezialist ist seit vielen Jahren als Prozessberater und Coach in IT-Projekten tätig. Schwerpunkte seiner aktuellen Arbeit sind neben der Frage, wie „agiles RE“ konkret aussieht, die Testbarkeit von Anforderungen und Abnahmekriterien, die Realisierung von „quality from the beginning“ sowie die Verbesserung agiler und nicht-agiler RE-Prozesse. Regelmäßige Publikationen und Konferenzvorträge runden sein Profil ab.