Conference Journal 2019

Interesting Insights
into Professional Practice

Papers of Lecturers at the Software Quality Days



Impressum

Publisher / Herausgeber

Software Quality Experts GmbH

Gewerbepark Urfahr 8

4040 Linz

https://www.software-quality-days.com/

info@software-quality-days.com

+43 5 0657-0

Published by Software Quality Days, January 2019

Disclaimer

The editor does not accept any liability for the information provided. The opinions expressed within the articles and contents herein do not necessarily express those of the editor. Only the authors are responsible for the content of their articles. The editor takes pains to observe the copyrights of the graphics, audio documents, video sequences and texts used, to use his own graphics, audio documents, video sequences and texts, or to make use of public domain graphics, audio documents, video sequences and texts. Any trademarks and trade names possibly protected by third parties and mentioned are subject to the provisions of the respectively applicable trademark law and the property rights of the respectively registered owners without restriction. It cannot be inferred from the mere mention of trademarks alone that these are not protected by third-party rights! The copyrights for published materials created by Software Quality Lab GmbH remain the exclusive rights of the author. Any reproduction or use of graphics and texts (also excerpts) without explicit permission is forbidden.

Haftungsausschluss

Der Herausgeber übernimmt keinerlei Gewähr für die bereitgestellten Informationen. Die in den Artikeln und Inhalten dargestellte Meinung stellt nicht notwendigerweise die Meinung des Herausgebers dar. Die Autoren sind alleine verantwortlich für den Inhalt ihrer Beiträge. Der Herausgeber ist bestrebt, in allen Publikationen die Urheberrechte der verwendeten Grafiken, Tondokumente, Videosequenzen und Texte zu beachten, von ihm selbst erstellte Grafiken, Tondokumente, Videosequenzen und Texte zu nutzen oder auf lizenzfreie Grafiken, Tondokumente, Videosequenzen und Texte zurückzugreifen. Alle genannten und ggf. durch Dritte geschützten Marken- und Warenzeichen unterliegen uneingeschränkt den Bestimmungen des jeweils gültigen Kennzeichenrechts und den Besitzrechten der jeweiligen eingetragenen Eigentümer. Allein aufgrund der bloßen Nennung ist nicht der Schluss zu ziehen, dass Markenzeichen nicht durch Rechte Dritter geschützt sind! Das Copyright für veröffentlichte, vom Autor selbst erstelltes Material bleibt allein beim Autor der Seiten. Eine Vervielfältigung oder Verwendung von Grafiken und Texten (aus auszugsweise) ist ohne ausdrückliche Zustimmung nicht gestattet.

Content

Article in this Conference Journal 4

A brief guideline for using this template

Firstname Lastname

Definition of Ready 5

Quality Gates für agile Requirements

Dipl.-Ing. Johannes Bergsmann

Separated but not divorced 11

Testing in distributed agile teams

Milena Petrovic

Article in this Conference Journal

A brief guideline for using this template

The abstract is about to raise the reader’s interest and give a stimulating insight into the article. Following questions can help you phrasing: What is the main message of the article? Which problems are being addressed? What do they mean to the reader?

In practice reports it’s helpful for the reader to mention the article’s basis (project, customer,…). Please spare references and footnotes and mind the maximum length of 600 characters (including spaces) for the abstract.

Title of the article

Use a short and concise title with max. 40 characters (including spaces) to describe the content of the article and encourage the readers to read. The Subtitle can be up to 100 characters (including spaces) long and should specify the title. We reserve the right to amend the title accordingly.

Article

The length of the whole article should be app. 8’000 to 40’000 characters (including spaces).

Please mind a balanced structure consisting introduction, description of the problem and its solution, consequences, experience with the solution and the résumé.

In order to structure your article you can use

sub headlines,

bullet points or

you can also use photos, tables, diagrams and

other images but please note that they must be identified and referenced in the text.

Paragraph numbering:

1. use the number defined paragraph numbering format
2. second point
3. etc.

If people are recognizable in the photos, they must have agreed to the release!

Graphic elements should be provided in print quality (300dpi, CMYK), they will not be revised before publication!

Article Examples

You will find 2 examples of articles below in this template – one in German language and another one in English language.

Fonts and formats

Please use the styles and fonts defined in this template. You can select the defined fonts in the menu bar.

Please don’t change the styles and fonts definition, the columns or section layout of the document.

The table of content is automatically generated based on some predefined styles.

If you want to import text into this template, please insert the text without using the original format and styles and then reformat the article with the predefined styles in this template.

Bibliography

Please only take sources into the bibliography that are actually being referenced in the text (no references in the footnotes).

It should essentially consist of printed sources. URLs can also be included if there is no adequate printed source available.

The bibliography should maintain maximum ten entries with recommended literature for further deepening. If you like to list web links please summarize them in a separate list.

Author



Firstname Lastname

Please briefly describe your role in the company or organisation (your company name can be published here), your specialization, business skills and interests with max. 300 characters (including spaces).

Please spare CVs and self-promotion.

name@domain.toplevel

Definition of Ready

Quality Gates für das agile Requirements-Engineering

Die „Definition of Ready“ für das Requirements-Engineering in agilen Vorgehensweisen ist als Eingangskriterium bzw. als Qualitäts-Checkliste für umzusetzende Requirements zu sehen. Nur Requirements, die diesen Qualitätslevel erfüllen, dürfen in die Umsetzungsplanung des Sprints genommen werden. Dieser Beitrag erklärt, was dabei zu beachten ist und wie eine Definition of Ready formuliert werden kann.

In agilen Vorgehensweisen gibt es eine Reihe von Quality Gates, die sicherstellen sollen, dass die Umsetzung der Anforderungen und die Abwicklung eines Projekts in ausreichender Qualität passiert. In „Abbildung 1“ sind einige Beispiele dafür angegeben.



Abbildung 1 - Quality Gates in Scrum

Auf der Ebene der Anforderungen sind insbesondere die Definition of Ready (DoR) sowie die Definition of Done (DoD) wichtig.

Die Definition of Ready legt die „Eingangskriterien“ für eine Anforderung in einem Sprint fest. Die Definition of Done legt die „Ausgangskriterien“ für eine Anforderung aus einem Sprint fest. Darüber hinaus können nach Bedarf und Umfeld noch beliebige weitere Quality Gates im agilen Umfeld angewendet werden. Im Wesentlichen sind es Checklisten, deren Punkte erfüllt sein müssen, damit im Prozess der nächste Schritt (mit dem geprüften Artefakt) ausgeführt werden kann.

Wichtig dabei ist, dass die Quality Gates eine Verbindlichkeit für das gesamte agile Team und eine angemessene Detailliertheit aufweisen, welche die einzelnen im Quality Gate enthaltenen Prüfpunkte auch klar verständlich macht, jedoch die Check-Liste nicht soweit überfrachtet, dass der Prozess nicht mehr effizient durchführbar ist.

Idealerweise werden die Quality Gates auch durch alle Prozessbeteiligten gemeinsam formuliert. Es kann jedoch auch notwendig sein, durch übergeordnete Instanzen (z.B. Prozess-Manager, Quality-Manager, Safety-Manager, Rechtsabteilung, etc.) Checkpunkte in die Quality Gates einzufügen, die nicht durch die Prozessbeteiligten vorgesehen waren, weil sie dem Team augenscheinlich keinen Nutzen bringen, aber im Sinn des Unternehmens unbedingt erforderlich sind. Auch solche Checkpunkte sind vom Team zu berücksichtigen und im Prozess einzuhalten.

Das Regelwerk der Quality Gates in agilen Prozessen ist ein sehr mächtiges Instrument, um den Prozess zu steuern. Es sollte mit Bedacht, Weitsicht und Angemessenheit angewendet werden und im Sinn einer kontinuierlichen Weiterentwicklung der Prozesse auch regelmäßig evaluiert und angepasst werden.

Definition of Ready (DoR) für Requirements

Bevor die Anforderungen vom Team geschätzt und für die Umsetzung eingeplant werden können, muss sichergestellt sein, dass sie verständlich, vollständig und detailliert genug beschrieben sind. Dies ist Aufgabe des Produkt Owners (PO) in Abstimmung mit dem Entwicklungsteam.

Neben der Verständlichkeit, Vollständigkeit und Detail­liertheit gibt es im Requirements Engineering eine Reihe von weiteren Qualitätskriterien, die erfüllt sein müssen, sowie Techniken zu deren Überprüfung.

Eine Voraussetzung für die Erstellung eines Produk­tinkrements ist, dass die dazu eingeplanten Anforderungen bereit für die Umsetzung sind.

Welche Qualitätskriterien eine Anforderung tatsächlich erfüllen muss und welche Analysen durchgeführt sein müssen, damit sie als „bereit für die Umsetzung“ bezeichnet werden darf, wird in einer „Definition of Ready“ festgelegt. Analog zur DoD werden darin diejenigen Kriterien explizit sichtbar gemacht, die eine Anforderung erfüllen muss, bevor sie für die Umsetzung in der nächsten Iteration akzeptiert wird.

Diese können helfen, Missverständnisse zwischen Team und Product Owner von vornherein zu vermeiden und die Effizienz weiter zu steigern.

Die DoR ist eine Liste mit Kriterien für die Aufnahme eines Requirements aus dem Product Backlog in den Sprint Backlog. Sie dient der Freigabe dieser Requirements für die Umsetzung.

Die Definition of Ready wird, wie die Definition of Done (DoD), zwischen PO und dem Umsetzungsteam vereinbart.

Wie bei der DoD, kann es auch bei der DoR sein, dass einzelne Kriterien nicht für jedes Requirement angewendet werden müssen. So ist es möglicherweise sinnvoll, für unterschiedliche Risikoklassen von Requirements auch unterschiedliche DoR-Kriterien anzuwenden.

Werden für ein Projekt oder eine agile Organisation Qualitäts­kriterien für Requirements definiert, so sollten diese jedoch auch in die DoR aufgenommen werden.

Die folgende Tabelle zeigt Beispiele für Kriterien, die in eine Definition of Ready aufgenommen werden können. Für jedes Kriterium ist beispielhaft auch angegeben, für welche Risikoklassedieses Kriterium bei der Requirements-Freigabe angewendet werden muss:

| **Prüfpunkt** | für Risiko |
| --- | --- |
| Ist die Anforderung vollständig beschrieben?Sind UI Masken ausreichend beschrieben?* Grafischer Maskenentwurf / Skizze
* Mögliche Aktionen
* Angezeigten Daten
* Regeln
* Berechtigungen zur Anzeige / Bearbeitung
 | H,M |
| Sind die externen Systemschnittstellen ausreichend beschrieben?* Datenobjekte und Felder inkl. möglicher Feldwerte
* Austauschformat
* Austauschprozess
* Erwartete Datenmengen
* Fehlerbehandlung
 | H,M,L |
| Sind Berichte ausreichend beschrieben?* Grafischer Entwurf / Skizze
* Angezeigte Daten
* Berechtigungen zur Anzeige
 | H |
| Sind nicht-funktionale Anforderungen definiert und geprüft? | H,M |
| Ist die Story verständlich und eindeutig? | H,M,L |
| Werden keine Universalquantoren (alle, nichts, etc.), Konjunktive, Passivkonstruktionen und Relativierungen verwendet?  | H,M,L |
| Wird die Satzschablone für User Stories eingehalten („Als ROLLE möchte ich AKTION, um NUTZEN“)? | H,M,L |
| Ist die Story bewertet? | H,M,L |
| Herrscht Übereinstimmung bei allen Stakeholdern, dass die Story wichtig für das Produkt ist? | H,M,L |
| Sind folgende Priorität, Business Value und Risiko bestimmt? | H,M,L |
| Sind die relevanten Akzeptanzkriterien definiert?Manche Akzeptanzkriterien ergeben sich erst im Zuge der Umsetzung, die wesentlichen müssen aber schon vorher definiert sein. | H,M,L |
| Sind die Risiken für die Story identifiziert und bewertet? | H,M |
| Sind die Risiken in einer Risikoliste dokumentiert?  | H,M,L |
| Sind für Risiken mit Wert > 15 Maßnahmen definiert? Wurden Risiken mit Wert > 20 an die GF gemeldet? | H,M |
| Wird die Story vom Tester als „testbar“ eingestuft? | H,M,L |
| Wurden die Testfälle erstellt?Basierend auf den Akzeptanzkriterien muss ein Testplan für die Prüfung der Story erstellt werden. | H,M |
| Wurde überlegt, wer die Story abnimmt und wie das Team sie demonstrieren kann?Oft wird eine Story nicht vom PO, sondern vom Kunden abgenommen. In diesem Fall muss dies vorher geplant werden. | H,M,L |
| Ist das Entwicklungssystem bereit für die Story?Kann aus personeller, technischer und Infrastruktursicht mit der Umsetzung der Story begonnen werden? | H,M,L |
| Erfüllt die Story die INVEST Kriterien?Die INVEST-Kriterien werden weiter hinten in diesem Beitrag beschrieben. | H,M,L |
| Sind alle Abhängigkeiten identifiziert?Abhängigkeiten können die Umsetzung einer Story verzögern und die Prüfung erschweren. | H |
| … |  |

Abbildung 2 - Beispielkriterien für eine Definition of Ready Checkliste

Die DoR kann über den Projektverlauf angepasst, erweitert oder auch reduziert werden, wenn sich das Team, der PO und die Stakeholder „eingeschwungen“ haben und die Zusammenarbeit so gut funktioniert, dass auch ohne laufendes Durchgehen einer Checkliste die Qualitäts­kriterien selbständig eingehalten werden.

Die DoR kann als Checkliste beim Taskboard aufliegen. Die Stories werden dann in der Iterationsplanung entsprechend gemeinsam bewertet und bei Erfüllung der Kriterien in den Sprint Backlog aufgenommen. Die DoR-Bewertungen werden in diesem Fall typischerweise nicht nieder­geschrieben. Durch das Aufnehmen der Story in den Sprint Backlog ist implizit dokumentiert, dass sich das ganze Team einig war, dass die Kriterien auch erfüllt sind.

In manchen Fällen wie z.B. bei sicherheitskritischer Software oder anderen haftungsrelevante Umgebungen, kann es notwendig sein, die DoR-Durchführung nach­vollziehbar zu dokumentieren. Um die DoR hier effizient und nachvollziehbar durchführen zu können, ist es sinnvoll, dies toolunterstützt zu machen z.B. mit einem entsprechenden Requirements-Management-Werkzeug oder einem elektronischen Taskboard.

Aufpassen muss man bei der DoR, dass diese nicht als Ausrede verwendet wird, um Diskussionen über User Stories vom Entwickler-Team fern zu halten, was dann genau den agilen Grundsätzen widersprechen würde.

Allgemeine Requirements-Qualitätskriterien

Nachfolgend werden einige allgemeine Qualitätskriterien für Requirements angeführt, die ebenfalls als Quelle für die in die DoR aufzunehmenden Kriterien herangezogen werden kann:

Qualitätskriterien sind unter anderem im Standard IEEE 830-1998 oder von IREB [1] definiert:

**Verständlich und eindeutig**

Alle Anforderungen müssen spätestens vor der Schätzung in einem Zustand sein, dass jeder im Team sie versteht. Sie müssen auch so eindeutig beschrieben sein, dass es keine unterschiedlichen Interpretationen über den Inhalt der Anforderung gibt. Hier kann es helfen, bestimmte mehrdeutige Worte und Satzkonstrukte zu vermeiden.

**Vollständig und angemessen**

Um Vollständigkeit zu erlangen, müssen alle Anforder­ungen aller relevanten Stakeholder gesammelt werden.

Damit keine Stakeholder übersehen werden, ist im Vorfeld eine Stakeholderanalyse sinnvoll.

Die Vollständigkeit kann durch gute Fragetechnik des Requirements-Erstellers, durch entsprechendes Hinter­grundwissen und Einblick in die zu spezifizierende Domäne gefördert werden. Leider ist es jedoch nicht möglich, die Vollständigkeit endgültig sicher zu stellen.

Angemessenheit ist relativ schwierig zu erlangen, weil es subjektiv sehr unterschiedlich sein kann, ob eine Anforderung angemessen ist – vor allem auch bei Betrachtung durch verschiedene Stakeholder. Was eventuell für den einen Stakeholder wichtig ist, ist für den anderen reine Zeit- und Geldverschwendung.

**Abgestimmt**

Der PO oder auch andere Auftraggeber sind alleine manchmal nicht in der Lage, eine Anforderung korrekt, vollständig und für die Umsetzer verständlich zu beschreiben. Daher sollten die Anforderungen mit allen relevanten Stakeholdern abgestimmt werden. Möglichkeiten dazu sind beispielsweise in Review­Meetings, im Sprint Planning Meeting aber auch jederzeit bei Bedarf vor der Umsetzung der jeweiligen Anforderungen.

**Gültig und aktuell**

Anforderungen können sich laufend ändern. Vor jeder Umsetzungsiteration muss daher der PO neu prüfen, ob die Anforderungen den aktuellen Stand der Kundenwünsche wiederspiegeln und noch gültig sind, bevor diese in einen Sprint für die Umsetzung aufge­nommen werden.

**Realisierbar**

Mit Anforderungen, die nicht umsetzbar sind, sollte sich das Team gar nicht weiter beschäftigen. Diese Anforderungen sollten durch den „DoR-Filter“ gar nicht erst in den Umsetzungssprint gelangen. Gegebenenfalls muss die Umsetzbarkeit eines Require­ments vorab durch eine Machbarkeitsanalyse geprüft werden.

**Bewertet**

Für die Release- und Iterationsplanung muss jede Anforderung bewertet sein. Ein wichtiges Eingangskriterium in den Umsetzungssprint ist die Bewertung, ob das Requirement die DoR erfüllt.

Weitere Bewertungsaspekte können z.B. Risiko, Dringlichkeit, Umsetzungsaufwand, etc. sein.

**Korrekt**

Eine Anforderung ist dann korrekt, wenn sie die **tatsächlichen** Bedürfnisse der Stakeholder wiederspiegelt.

**Konsistent**

Die Anforderungen sollen in sich und über alle Anforderungen hinweg konsistent, also widerspruchsfrei sein.

Da das Product Backlog lediglich die noch offenen Anforderungen enthält und somit keine Gesamtspezifikation existiert, kann mit dem Backlog alleine die Widerspruchsfreiheit nicht geprüft werden. Des Weiteren ist der Detaillierungsgrad weiter in der Zukunft liegender Anforderungen recht grob und diese damit kaum auf Konsistenz prüfbar. Die Prüfung auf Konsistenz wird daher durch den PO und das Team gemeinsam durchgeführt. Durch die enge Zusammenarbeit zwischen PO (fachlich) und Team (technisch) erfolgt ein ständiger Austausch und ein ständiges prüfen, ob die gerade in Umsetzung befindlichen Stories auch mit dem bisher Erreichten zusammenpassen. Bei der Konsistenzprüfung werden nicht nur die spezifizierten Anforderungen berücksichtigt, sondern auch das implizite Wissen in den Köpfen des PO und des Teams über die noch ungenau spezifizierten Anforderungen im Backlog sowie über bereits fertige Softwareteile.

**Prüfbar**

Jede Anforderung, die in einen Umsetzungssprint aufgenommen wird, muss so beschrieben sein, dass am fertigen Produktinkrement geprüft werden kann, ob sie auch richtig umgesetzt wurde. Die Kriterien sollten vor der jeweiligen Iteration definiert werden durch:

Akzeptanzkriterien zu jeder Anforderung

Definition of Done

Idealerweise Testfälle oder konkrete Beispiele

Zumindest die Akzeptanzkriterien und die Definition of Done müssen immer vorhanden sein.

Wenn eine nachhaltig qualitätsorientierte Entwicklung angestrebt wird, kann nur dringend empfohlen werden, die Prinzipien des testgetriebenen Vorgehens zu berücksichtigen und die Definition von passenden Testfällen als verbindliches Kriterium für die Umsetzungs-Freigabe einer Anforderung festzulegen.

Dabei geht es nicht darum, dass alle theoretisch denkbaren Abnahmekriterien und Testfälle vorab schon festgelegt werden. Es geht hier um diejenigen Testfälle, die das prüfen, was dem Kunden wichtig ist (z.B. ob die Funktion auch den korrekten Wert liefert) und das, was nach der Erfahrung guter Tester auch ohne expliziten Kundenwunsch sinnvollerweise getestet werden sollte (z.B. grenzwertige Eingaben, die erfahrungsgemäß oft zu Fehlern führen oder risikoreiche Eigenschaften oder Anwendungen der Anforderung, auch wenn sie nicht unmittelbar im Anwenderfokus stehen).

Es gilt in der Software-Entwicklung der Grundsatz: Solange man für eine Anforderung nicht genau weiß, wie man sie später abnehmen und testen wird, ist sie nicht reif für die Umsetzung!

*Beispiel für nicht prüfbare Anforderungsdefinition:*

*Ein Stakeholder hat als grobe Anforderung definiert: „Als Mitarbeiter möchte ich meine Zeiten erfassen können“.*

*Er wird dann vom PO gefragt, wie er das testen und abnehmen möchte.*

*Wenn als Antwort kommt: „Weiß ich nicht.“, dann kann es sein, dass der Stakeholder noch keine Zeit hatte, sich detailliertere Gedanken zu machen. Der PO sollte hier so lange nachfragen bzw. Testfälle mit dem Stakeholder festlegen, bis diesem klar ist, was er eigentlich möchte.*

*Der andere Fall ist, dass der Stakeholder bewusst nicht genau spezifizieren möchte, ganz nach dem Motto „Ist mir egal, ich habe keine Zeit darüber nachzudenken, liefern Sie mir eine brauchbare Lösung!“. Dies ist ein kritischer Fall für den PO. Damit es nachträglich nicht zu Ärger kommt, muss der PO dem Stakeholder klar machen, dass er bei dieser ungenauen Anforderungsspezifikation am Ende dann auch die Abnahme nicht verweigern kann, wenn die Mitarbeiter nur irgendwie die Zeiten erfassen können, dies jedoch gar nicht seinen Vorstellungen entspricht.*

**Verfolgbar:**

Das Kriterium Verfolgbarkeit (Traceability) definiert die Verknüpfung von Anforderungen mit anderen Artefakten, wie z.B. Ziele, Architekturmodellen, Testfällen, Quellcode, Normen, externe Artefakte, usw.

Diese Verweise zu anderen Artefakten können nur mit einem professionellen Anforderungsmanagement- bzw. ALM-Werkzeug sinnvoll verwalten werden.

**Zusätzlich zu den Qualitätskriterien in IEEE 830-1998 und IREB können noch weitere Eigenschaften als Kriterien in der DoR definiert werden:**

**Relevante Stakeholder:**

Vor allem für den PO ist es wichtig zu wissen, welche Kunden und Stakeholder an einer Anforderung interessiert sind bzw. diese definiert haben.

Es sollte für jede Anforderung klar sein:

Wer hat sich diese Anforderung gewünscht?

Wer trifft die Letztentscheidung bei fachlichen Fragen?

Wer nimmt die Story ab?

Wer soll zusätzlich informiert werden bzw. ist betroffen?

Wer ist die Zielgruppe für die Anforderung?

Da es möglich und auch oft üblich ist, dass Anforderungen stellvertretend von jemand anderem formuliert werden, sollte bei jeder Anforderung jedenfalls noch der sogenannte „Anforderungs­provider“ – also derjenige, der diese Anforderung explizit haben möchte - angegeben werden, damit man bei Fragen weiß, an wen man sich wenden kann. Anstelle von Gruppenbezeichnungen (z.B. „Team A“, „die IT“, „Abteilung X“) sollten immer konkrete Namen (z.B. „Hans Mair“, „Franz Schmidt“) angegeben werden, damit der Stakeholder bzw. Anforderungs­provider „greifbar“ bleibt.

**Identifizierbarkeit:**

Die Referenzierung einer Anforderung wird enorm erleichtert, wenn diese eine eindeutige ID hat. Im einfachsten Fall ist dies eine Nummer, es können aber auch beliebig strukturierte Kennzeichnungen verwendet werden.

Wichtig dabei ist nur, dass die Kennzeichnung innerhalb des Bereichs, in dem referenziert werden soll (z.B. Task-Board, gesamter Backlog, gesamte Produk­tent­wicklung mit allen Artefakten oder auch gesamtes Unternehmen oder weltweit) eindeutig ist.

****INVEST Qualitätskriterien****

Speziell für die Beurteilung der Qualität von User Stories hat sich ein Set von Qualitätskriterien herauskristallisiert, die unter dem Akronym INVEST bekannt geworden sind [2]**:**

**Independent:** Die Stories sind unabhängig voneinander. Jede Story kann als eigenständiges Paket geschätzt, geplant und umgesetzt werden.

**Negotiable:** Die Stories sind verhandelbar (und verhandelt). Sie stellen keinen fixen Vertrag dar. Stellt sich im Zuge der Umsetzung heraus, dass ein Detail anders einfacher oder benutzerfreundlicher umgesetzt werden kann, so wird dies mit dem PO abgestimmt und dann geändert.

**Valuable:** Jede Story muss einen erkennbaren Wert für den Kunden liefern.

**Estimatable:** Der Aufwand für die Story ist zumindest relativ schätzbar (und geschätzt). Dies ist notwendig, um die Story planen zu können.

**Small:** Jede Story muss innerhalb einer einzigen Iteration umsetzbar sein. Sie sollte daher, je nach Sprintlänge, eine Größe von maximal fünf Umsetzungstagen haben.

**Testable:** Für jede Story sind klare Kriterien festgelegt, wann sie fertig ist. Diese Kriterien sind in den Akzeptanzkriterien bei der Story selbst und in der „Definition of Done“ festge­schrie­ben.

****Qualität der Kriterien für die DoR****

**Wenn nun Kriterien in die DoR aufgenommen werden, sollte darauf geachtet werden, dass diese „SMART“ sind:**

**Spezifisch:** Die DoR Kriterien sollen eindeutig und so präzise wie möglich definiert sein und nicht vage.

**Messbar:** Die Kriterien müssen messbar bzw. testbar sein.

**Akzeptiert:** Die DoR Kriterien müssen angemessen sein und von den Betroffenen akzeptiert werden.

**Realistisch:** Die Einhaltung bzw. Erreichung muss möglich sein.

**Testbar:** Es muss klar sein, wie getestet und abgenommen wird.

Es kann auch sein, dass einzelne DoR Elemente nicht für jede Anforderung angewendet werden. So ist es mögli­cher­weise sinnvoll, je nach Risiko einer Anforderung unterschiedliche DoR-Kriterien zu haben. Damit man in die DoR nicht zu wenig, aber auch nicht zu viele oder vielleicht sogar unsinnige Kriterien aufnimmt, sollte man sich vorab bei der Erstellung der DoR schon intensiv Gedanken machen.

Unterstützt werden kann dies durch ein sogenanntes „Done Thinking Grid“ (Abb. 3). Dies ist eine Technik, mit der auf einem Board Ideen zu sinnvollen Done-Kriterien gesammelt werden. Zur Strukturierung können evtl. schon Themenbereiche und Ebenen vorgegeben werden (z.B. als „Swimlanes“).



Abbildung 3 - Beispiel für „Done Thinking Grid“

In der Literatur und im Internet sind oft unklare Begriffsdefinition bzw. Beispiele für DoR’s, die dazu verleiten, DoD‘s zu oberflächlich oder ungeeignet zu erstellen.

Diese oberflächlichen Beispiele führen in der Praxis zu dem Problem, dass unterschiedliche Stakeholder, Entwickler bzw. Teams oder der Kunde dies unterschiedlich interpretieren.

DoD‘s sollten nur Qualitätskriterien an den Requirements-Prozess und allgemeine Qualitätskriterien für die Require­ments selbst enthalten. Sie sollen keine Testfall­beschreibungen oder inhaltlichen Abnahmekriterien enthalten oder diese ersetzen.

Ebenso sollten Kriterien, die spezifisch zu einer Anforderung gehören, in deren Akzeptanzkriterien aufgenommen werden und nur übergreifende Kriterien in die DoR. Für eine einfache und klare Anwendung der Definition of Done sollte vermieden werden, Testfälle und DoD-Kriterien zu vermischen!

Tests und inhaltliche Anforderungen jeglicher Art sollten in den jeweiligen Anforderungs- und Test-Artefakten definiert werden und nicht in einer DoR.

Die DoD soll ausschließlich qualitative Kriterien und keine inhaltlichen Anforderungskriterien an das zu erstellende System oder einzelne Artefakte enthalten!

Folgende Tipps helfen auch dabei, DoD‘s passend zu erstellen:

Ein **einziger** DoR-Eintrag ist in den meisten Fällen zu wenig!

**Jedes Requirements-Artefakt durchläuft verschiedene Phasen / Ebenen. Es sollten daher verschiedene DoR-Checkliste für jede dieser Phasen / Ebenen erstellt werden.**

**„Think twice“: Nach dem ersten DoR-Entwurf sollten man nach einer gewissen Zeit selbst nochmal darüber nachdenken und die DoR von anderen Personen reviewen lassen.**

**Regelmäßige Anpassung der DoR-Checkliste.**

**Berücksichtigen der Projekt-Kritikalität und des Projekt-Umfelds.**

DoR‘s sollten so in den Prozess integriert sein, dass sie möglichst zwingend eingehalten werden müssen, z.B. durch …

Vier-Augen-Prinzip in verschiedenen Prozessphasen

Die DoR nicht alleine abhaken und das Requirement alleine freigeben (Das Team prüft und gibt frei!).

Der unternehmensweite Prozessverantwortliche sollte die Einhaltung der DoR-Anwendung übergreifend prüfen.

Der projektbegleitende Quality-Manager sollte die Anwendung der DoR in einem spezifischen Projekt prüfen.

Die DoR-Checklisten sollte – soweit dies möglich ist - in die verwendeten Tools eingebaut werden.

Ein Workflow-Tool kann verwendet werden, um die DoR dadurch direkt in die Prozess-Steuerung zu integrieren.

Eine Prüfung und (Selbst-)Kontrolle bei der DoR-Umsetzung ist wichtig, da die Anwendung viel Aufwand bedeuten kann und damit zu rechnen ist, dass die Durchführungsdisziplin bei fehlender Prozess-Absicherung und Prüfung nachlässt.Damit DoR‘s auch gelebt werden, ist es wichtig, diese *vor* Projektbeginn unter Einbeziehung aller relevanten Beteiligten zu definieren. Im Laufe des Projekts sollte das Team die DoR‘s in den Retrospektiven immer wieder kritisch hinterfragen und sie bei Bedarf weiter anpassen.

Ziel ist es, kontinuierlich besser zu werden und die Qualitätsansprüche in den DoR‘s laufend anzuheben!

Bibliography

**Bergsmann, Unterauer,** Requirements Engineering für die agile Softwareentwicklung**, dPunkt Verlag , 2014**

**IEEE 830-1998,** Recommended Practice for Software Requirements Specifications**,** IREB, International Requirements Engineering Board <http://www.certified-re.de/>

**Leffingwell**, Dean, *Agile Software Requirements,* Addison Wesley，2011

Quelle für **Done-Thinking-Grid**:

<http://www.scrumalliance.org/community/articles/2008/september/definition-of-done-a-reference>

Autor



Dipl.-Ing. Johannes Bergsmann

Geschäftsführender Gesellschafter
Berater & Trainer bei Software Quality Lab

Johannes.Bergsmann@software-quality-lab.com

Separated but not divorced

Testing in distributed agile teams

With a great boom of agile way of leading a project, requirements are getting less and less detailed. Having just a broaden draft of documentation just in time for coding makes our testing work pretty mess. Tester has to be proactive and wise to ask the right question the right people in the right moment and that task is getting harder when people are on different locations. The real challenge is how to keep on track with requirements and changes? How to lead a testing team from remote location? The answer to that is really not easy and for sure is not one thing.

In the last decade great investments were done to convert national project to global project. We are spreading our work across towns, countries and continents. There are many reasons for that. The most important are skill set availability, cost and complexity, resources constrains, government restrictions…On the other hand we endorse agile way of working. We want to be efficient, quick and proactive. Researches show that more than 80% of agile teams are distributed across multiple locations. It is widely believed that distributed work impose many challenges that normally don’t exist in collocated teams. Most common challenges would be delayed feedback, lack of communication, less shared project awareness, lack of trust between sites and maintaining good quality. All of that have an effect on quality. Test teams are responsible for maintaining good quality and distributed agile teams bring up the challenge whether we are able to persist under pressure. Test teams are in most of the cases distributed. It is very common to have test engineers as contractors from other side of the world. How does this affect overall progress? After all, the adage says, “Out of sight out of mind”, but the quality cannot be ignored. Hey, we are separated but for sure not divorced!

Agile brings in big changes

Agile way of working have big influence on project organisation and all its artefacts. Its influence can be felt on documentation, planning meetings, team roles and responsibilities, project organisation… Since testing is very important and since it has to be included from the project start up, all changes are reflecting on testing.

Documentation itself suffers the most. We try to keep it simple but also we have to have enough for good work. This is what makes testing difficult. When test team is distributed on remote location many questions will appear bringing up challenges such as when to ask, how to ask, whom to ask? One could never know what is happening away from his eyes. Is the contact person free? Is he/she in the mood for conversation? Bottom line, do we interrupt any important work? In this kind of situations tester has to be wise, assertive but balanced, proactive. He/she must invest much more effort than located engineers to rectify gaps in the documentation.

Testing cannot be performed without documentation, that is clear. We can’t verify something that is not written and confirmed. And what happens in agile teams? In most of the cases on rapid agile projects documentation is getting less and less detailed and very often it happens that changes are not noted on time. Many things change and only few are documented. If you are working on some remote location it is very common that you won’t be in a communication loop and you won’t be notified about change.

Creating detailed requirements and system specifications in advance may seem less risky, but this actually creates a false sense of security for the remote teams. The right way to do it is creating less specification up front and then, as project is evolving, through good cooperation get to the details. Problem with the documentation and the inability to follow development, changes and plans is just one of the problems with which remote test team struggles. There are other problems too.

Choosing the right approach

Different approaches in solving problems can be applied on differently organized distributed teams depending on which teams are distributed, on how many locations the product is developed and cultural differences.

During my three years of work on a project developed in SCRUM framework 3 countries were involved together with 4 languages. 4 major risks that affect the quality of our the product were:

Loss of communication richness

Lack of coordination

Geographic dispersion

Cultural differences

Testing team was part of this big project as completely equal member and final say “Go” or “No go” was up to us so quality was the most important part. We had to be in every single pore, on all meetings, every plan or scheduler and we were distributed from the main team on other location, in other country. Quite a challenge!

We had following organization:

Main team

Distributed developers

Distributed testers

The biggest team was on main location. All leads, SCRUM master, product owner and stakeholders were together on one location and developers and test team were distributed on 2 different locations. We had problems in different areas that reflected our work and made our work pretty a mess. Traveling on-site was one of the solutions but it cost us a lot of time, and time is money. Main problems and most appropriate solutions for us were:

**Stand up meetings:** with small investments we managed to have daily stand ups over video calls and efficient tools, It was simple and easy to implement and it was of great help to us to feel like part of the team. Still we had problems - awareness of people to use this. They had to be persuaded to adhere to an agreement to schedule the meetings and use video calls. Not everyone accept changes so easily so it took a lot of time and effort to make everyone understand how this is important and yet easy to implement.

**Planning meetings:** planning meetings were very slaw and not so efficient. We needed a lot of time to prepare, get into the plan, actually plan with others and make decisions. Since we were not efficient and didn’t have expected progress we decided to split planning into two planning meetings. The purpose of the main meeting was to talk about milestones, general plan, dividing tasks. It was preferred to have it on-site. Second part of the meeting was held internally with teams, on their location. This way team could analyse problems in details, divide tasks and solve small problems.

**Missing roles:** imagine a basketball team on one building and it’s coach in another one. There is no video call that could help them coordinate and be successful. Every team has to be present on the main location in some way to keep track of what are the requirements, needs and how situation is developing. Since we weren’t co-located our solution was to have deputies on each remote location. Lead (located) and its deputy (distributed) were in touch all the time discussing all topics. Their bond was the crucial for connecting people and teams.

Building trust: Nothing is more important than have trust in your co-workers. Trust is the crucial element of a hyper-productive teams and it’s not easy to gain it. Gaining trust is a challenge when team members are distributed across different time zones, cultures, and environments, and when they also face communication, language, technical alignment, and project management issues. When a team doesn’t have a minimum level of trust, it’s more difficult to deal with challenges when they appear. We all agree testing is important but we can agree also that testing is not as welcomed as we want it to be. If people know each other and if they have trust in each other than reporting ones mistakes can be accepted with more tolerance and patience. If people are not working closely together than they will not have enough trust and won’t feel the need to bring team spirit and positive energy. Than it is often easy to blame and criticize distributed teams. From my perspective, building trust between distributed teams is the biggest challenge because when trust is strong, team members are able to work through the most difficult issues and they often create innovative solutions.

Research done on our project showed that there aren’t almost any relationship between geographic distance of remote testing team and quality and defect density.

No matter in what kind of organization team you work on within distributed teams, the way we communicate and build trust is the crucial component required to achieve successful project with good quality. Communication will help us see each other as a team players who wants the same – to reach schedule, maintain quality and lower cost. Even though The Agile Manifesto states that “the most efficient and effective method of conveying information to and within a development team is face-to-face conversation” working in a distributed teams can be efficient and test team can feel like part of it. We for sure have to invest a lot of energy to keep us on track but others can also help.

Author



Milena Petrovic

Software engineer working for Comtrade Solutions Engineering.

Her journey started with development and moved forward to QA in agile projects in 2008. After years of testing she became lead engineer on different projects for world’s leading gaming companies.

milena.petrovic@comtrade.com