„Definition of Done“ – Quality-Gates im agilen Umfeld

Die „Definition of Done“ ist in agilen Vorgehensweisen die Fertigstellungs-Checkliste für die von den Entwicklern umgesetzten User Stories oder anderen Anforderungen. Nur umgesetzte Anforderungen, die den in der DoD angegebenen Qualitätslevel erfüllen, dürfen als „fertig“ gekennzeichnet und in das nächste Produkt-Inkrement übernommen werden. Dieser Beitrag erklärt, was bei der Definition of Done zu beachten ist und wie diese formuliert werden kann.

Motivation

Immer wieder kommt es in Softwareprojekten zu Fehlern und Unmut, weil nicht klar geregelt ist, wann etwas wirklich „fertig“ ist. Für einen Entwickler ist eine Anforderung vielleicht schon fertig, wenn er den Code geschrieben und erfolgreich kompiliert hat. Sieht sich dann der Product Owner die umgesetzte Funktion am Testsystem an, stellt er oft fest, dass noch viele Fehler vorhanden sind und die Benutzerbarkeit nicht wie gewünscht gegeben ist.

Für den Product Owner ist die Funktion erst fertig, wenn sie fehlerfrei einsetzbar und für den Anwender gut bedienbar ist. Aus solchen unterschiedlichen Sichtweisen resultieren in vielen Projekten Probleme und Unstimmigkeiten im Team. Ursache ist dabei eine fehlende explizite, für alle verbindliche und klare Definition, wann etwas „fertig“ ist.

Die einfachste Lösung für dieses Problem ist eine explizite Definition, wann ein Backlog Item auf „fertig“ gesetzt werden darf. Ken Schwaber und Jeff Sutherland schlagen dazu eine „Definition of Done“ (DoD) vor, damit alle „ein gemeinsames Verständnis haben, was es bedeutet, wenn eine Arbeit abgeschlossen ist“ [1].

Definition of Done (DoD)

Die Definition of Done ist eine Checkliste, die zwei Sichten enthalten kann:

- Die Produktsicht umfasst Checklistenkriterien, die darauf abzielen, einen bestimmten Qualitätslevel der Eigenschaften eines umgesetzten Backlog-Items zu erfüllen.
- Die Prozesssicht definiert Kriterien für alle Aufgaben und Tätigkeiten, die dabei (erfolgreich) erledigt sein müssen.

Bei der Planung der Iterationen und der Schätzung der Backlog Items muss das Team nicht nur die reinen Programmieraufwände berücksichtigen, sondern die Aufwände für die gesamte Abarbeitung der DoD (also z.B. auch fertig ausgeführte Test, Dokumentation, etc.).

Frustvermeidung durch Definition of Done

Wie die Definition of Done dabei hilft, Frust bei den Beteiligten einer Softwareentwicklung zu vermeiden:

Ich kann mich noch gut an meine Zeit als Produktmanager einer großen Webapplikation erinnern. An viele Diskussionen mit den Entwicklern, ob denn jetzt eine Funktion fertig ist oder nicht. Nein, war sie meistens nicht. Irgendwas hat immer gefehlt. Entweder wurde zu wenig getestet (manchmal auch gar nicht), oder die Systemdokumentation war nicht aktualisiert oder es fehlte einfach ein Teil der Funktion selbst. Sehr ärgerlich war das immer.

Nach einigen Monaten des Hoffens und Diskutierens hatten wir genug. Wir erstellten gemeinsam im Team eine einfache Liste, auf der wir vereinbarten, wann etwas als "fertig" bezeichnet werden durfte. Heute würde man zu der Liste "Definition of Done" und "Definition of Ready" sagen. Damals war es einfach nur unsere Abnahmecheckliste.

Was damals draufstand? Als erstes einmal "Für jede Funktion muss zumindest der Standardablauf einmal getestet worden sein" und "Bei jeder Funktion muss der Produktmanager vorab spezifizieren, wie er sie für die Abnahme testen wird". Schon alleine diese beiden Punkte haben mein Leben als Produktmanager und das meiner Entwickler dramatisch entspannt. Auf einmal war klar, was ich für die Abnahme erwarte und was das Team dafür braucht.

Und genau darum geht es in der "Definition of Done". Was müssen wir alles tun, damit wir am Ende stolz "fertig!" verkünden dürfen ohne dass es beim ersten Klick gleich scheppert.

Markus Unterauer

Berater, Trainer

Der Quality Knowledge Letter ist ein periodisches Informationsmedium von Software Quality Lab und dessen Partnern mit dem Schwerpunkt SW-Qualität.

Weitere Infos zu diesem und anderen Themen finden Sie auf www.software-quality-lab.com

Eine DoD ist nicht nur zur Bewertung der Fertigstellung eines einzelnen Backlog Items sinnvoll, sondern kann auf mehreren Ebenen angewendet werden, wie beispielsweise:

- **DoD für eine Anforderung** (Story oder anderes Backlog-Item). Die Bewertung der Einhaltung dieser DoD kann direkt nach Umsetzung eines einzelnen Backlog Items oder am Ende eines Sprints erfolgen.
- **DoD für einen Sprint**: Diese ist anforderungsübergreifend und bezieht sich auf den gesamten Sprint Backlog und alle sonst relevanten Artefakte und Tätigkeiten. Die Bewertung der Einhaltung der DoD erfolgt am Ende eines Sprints.
- **DoD für ein Release** (Auslieferung an den Kunden): Diese ist anforderungsübergreifend und gilt auch über mehrere Sprints hinweg für eine tatsächlich ausgelieferte Version des Systems. Die Bewertung erfolgt vor dem jeweiligen Release – also vor der tatsächlichen Auslieferung an den Kunden.

Je nach Umfeld, Anwendungsbereich oder Zielgruppe können weitere DoD-Ebenen oder eine zusätzliche Granularität sinnvoll sein:

- DoD für die Codierung: z.B. Kriterien, die definieren, wann ein Programmierer mit dem Code fertig ist und in die Quellcodeverwaltung einchecken darf, oder Forderungen bezüglich Unit-Test-Abdeckung, Code-Dokumentation, Code-Review, etc.
- DoD für Design / Architektur: z.B. eine Checkliste für Systemdesign, Architekturkriterien, Testbarkeit, Wartbarkeit, etc.
- DoD für die Testdurchführung z.B. Anforderungen an die Tests oder Testende-Kriterien, die definieren, wann man zu testen aufhört.
- DoD für den Roll-Out z.B. eine Checkliste für Verteilung, Inbetriebnahme, Konfiguration, etc.

Der Vorteil bei der Aufteilung der DoD auf mehrere Checklisten ist, dass bei der Durchführung nicht mehr eine sehr lange Checkliste herangezogen wird, unter der dann alle im Team stöhnen und die dann eventuell aufgrund der Länge nur oberflächlich oder lückenhaft durchgegangen wird. Der Aufwand wird vielmehr auf mehrere kleine Schritte aufgeteilt.

Kriterien in der DoD

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

- **Spezifisch**: Die DoD Kriterien sollen eindeutig und so präzise wie möglich definiert sein und nicht vage.
- **Messbar**: Die Kriterien müssen messbar bzw. testbar sein.
- **Akzeptiert**: Die DoD 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 DoD Elemente nicht für jede Anforderung angewendet werden. So ist es möglicherweise sinnvoll, je nach Risiko einer Anforderung unterschiedliche DoD-Kriterien zu haben.

Damit man in die DoD nicht zu wenig, aber auch nicht zu viele oder vielleicht sogar unnötige Kriterien aufnimmt, sollte man sich vorab bei der Erstellung der DoD schon intensiv Gedanken machen. Unterstützt werden kann dies durch ein sogenanntes „Done Thinking Grid“. Dies ist eine Technik, mit der oft auf einem Board, Ideen zu sinnvollen Done-Kriterien gesammelt werden. Zur Strukturierung können evtl. schon Themenbereiche und Ebenen vorgegeben werden (z.B. als „Swimlanes“):

- Requirements / Tasks
- Architecture / Design / Coding
- Testing
- Sprint / Release / Roll-OutProzess / Tools

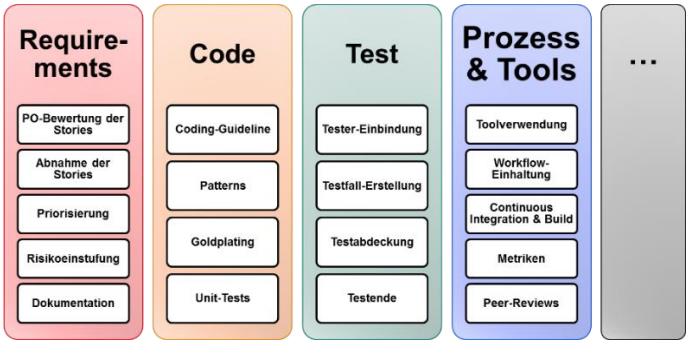

Abb. 3 Beispiel für „Done Thinking Grid“

DoD in der Praxis

Die in der Literatur und im Internet oft unklare Begriffsdefinition der DoD kann dazu verleiten, DoDs zu oberflächlich oder ungeeignet zu erstellen. Es werden oft zu einfache Beispiele für DoDs genannt:

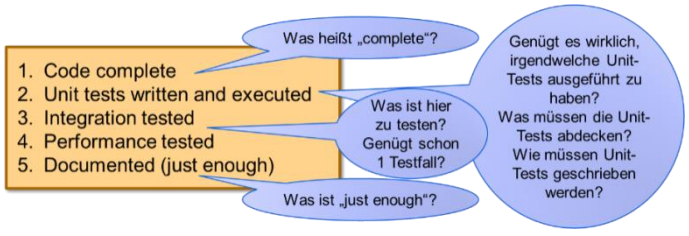

Abb. 4 Schlechtes Beispiel für eine DoD

Diese oberflächlichen Beispiele führen in der Praxis zu dem Problem, dass unterschiedliche Entwickler bzw. Teams oder der Kunde dies unterschiedlich interpretieren. Das Team meint eventuell, dass es fertig ist – tatsächlich sind dann aber noch viele Fragen offen, wenn der Kunde zur Abnahme hinzugezogen wird.

Vollständiger Knowledge Letter Zugang

Wir freuen uns, dass Sie an diesem Thema Interesse haben und den Knowledge Letter von Software Quality Lab bis hierher gelesen haben.

Dieser Knowledge Letter ist eine Vorschau (gekürzte Version des gesamten Artikels).

Wenn Sie den ungekürzten Knowledge Letter lesen möchten, registrieren Sie sich bitte unter
http://www.software-quality-lab.com/download/knowledge-letter/anfrage-knowledge-letter/

Sie erhalten nach der Registrierung vollen Zugang zu allen bisherigen Knowledge Lettern von Software Quality Lab und erhalten automatisch künftige Knowledge Letter per E-Mail.