Pitch für cards+

Robert Bruckbauer

Version 2.2.0, 2020/09/06: cards+ asciidoc book

Inhalt

1. Vorwort

Ich habe kürzlich an einer persönlichen Weiterbildung teilgenommen, bei der ich von einem Profi (u.a. Coach bei der PULS4-Show 2 Minuten 2 Millionen) in die Geheimnisse des *pitchens* eingeweiht wurden. Ich habe gelernt, dass die ersten Sätze in einem Pitch entscheidend sind. Sie müssen Interesse wecken, die Zuhörer neugierig machen. Als nächstes sollte das Problem dargestellt werden und "verlängert" werden. Verlängern heißt, aus dem ursprünglichen Problem (engl. root cause) Konsequenzen abzuleiten, durchaus mehrstufig. Erst dann präsentiert man seine Lösung. Gelerntes soll man ja sofort anwenden, um es zu vertiefen. Darum versuche ich spontan einen Pitch für `cards+`.

2. Einstieg

Eine Software besteht aus Quelltexten, Testplänen und weiteren dokumentierten Informationen. Quelltext klar, das ist das Produkt. Testpläne in einer Testpyramide sichern die Qualität des Produktes. Weitere dokumentierte Informationen sind Verträge, Aufträge, Anforderungen, User-Storys, Studien, Grob- und Fachkonzepte, Analysen, Handbücher, Online-Hilfen u.v.m. In Projekten mangelt es in der Regel nicht an Dokumentation. Wollen wir Software erfolgreich über einen längeren Zeitraum betreiben, dann müssen unsere Entwickler kontinuierlich Fehler beheben, Patches und Updates einspielen und neue Anforderungen umsetzen. Dazu müssen die Entwickler nicht nur den Quelltext verstehen, sondern das Produkt als Ganzes.

3. Problem

Durch Fluktuation bei den interessierten Parteien oder in der Projektorganisation geht uns Wissen verloren. Nun, sie werden sagen, dass alles Wissen im Quelltext steckt. Ja, Wissen steckt im Quelltext. Der Quelltext beantwortet die Frage, wie etwas gelöst wurde. Testpläne beantworten die Frage, was das Produkt nachweislich leistet. Aber die Frage, warum das Produkt gebraucht wird, wer es einsetzt und welchen Zweck er dabei verfolgt, das kann nur durch die weiteren dokumentierten Informationen beantwortet werden. Und das ist der Kern des Problems: Diese Informationen sind in der Regel veraltet, widersprüchlich, falsch.

Ein Fachkonzept kann formal völlig in Ordnung sein, und war wahrscheinlich zum Zeitpunkt der Freigabe auch korrekt. Aber durch jede Fehlerbehebung, Umsetzung einer Anforderungen oder Änderung im Design der Software entsteht eine Abweichung vom ursprünglichen Konzept. Software wird kontinuierlich verändert. Konzept und Software passen sehr bald nicht mehr zusammen. Fachkonzepte werden mit der Zeit unbrauchbar, wenn sie nicht aktualisiert werden. Und das passiert sehr selten. Die Gründe sind vielfältig. Aber das ist eine ganz andere Geschichte.

Ein Handbuch kann einem Entwickler helfen, das Produkt zu verstehen. Aber ein Handbuch wird aus der Perspektive der Nutzer geschrieben. Ein Handbuch sieht die Software immer als Blackbox, von außen. Ein Handbuch kann daher einem Entwickler nichts über die Gründe für den Entwurf der inneren Struktur der Software sagen.

Die Liste der dokumentierten Informationen, die einem Entwickler **nicht** helfen, ist sehr, sehr lang. Frustrierend lang. Falsche Dokumentation birgt zusätzlich die Gefahr, dass unsere Entwickler in die Irre geführt werden. Und das kostet Geld.

4. Lösung

Die Lösung ist eigentlich ganz einfach. Wir wollen eine Produktdokumentation, auf die wir uns verlassen können. Mit "wir" ist dabei die gesamte Projektorganisation gemeint, inklusive der interessierten Parteien des Auftraggebers. Neben dem Quelltext und den Testplänen des Produktes muss von Anfang an auch die Dokumentation für das Produkt inkrementell erarbeitet und mit dem Fortschritt der Entwicklung kontinuierlich verbessert werden. Dokumente müssen wie Quelltext betrachtet und versioniert werden. Dokumente müssen strukturiert sein. Sie müssen prüfbar sein. Ihre Qualität muss messbar sein. Im Grunde gelten sehr viele Praktiken der Entwickler auch für Autoren einer Dokumentation: KISS, DRY, INVEST, die Pfadfinderregel. Das gemeine ist, dass Entwickler schon sehr lange geeignete Strategien haben, um die Komplexität von Software in den Griff zu kriegen. Sie wählen Strukturen, die besser testbar sind als andere. Sie modularisieren. Sie delegieren. Sie kapseln.

Mit `cards+` gibt es ein Konzept, das solche Strategien auch für eine Produktdokumentation bietet. `cards+` macht es möglich, die Ergebnisse aus der Analyse (z.B. mit Domain-Driven Design) direkt in ein Wiki zu übernehmen. `cards+` ist ein in sich geschlossenes Ökosystem von Bausteinen. Jeder Baustein hat einen klaren Zweck, eine prüfbare Struktur und einen für den Inhalt verantwortlichen Autor. Die Bausteine sind aufeinander abgestimmt mit dem Ziel, jede Information nur einmal zu erfassen. Im Wiki ist jeder Baustein eine Seite. Bausteine bilden Hierarchien (z.B. Topic, Epic und Case) oder werden miteinander verknüpft (z.B. Begriffe aus dem Glossar, die Lösung in einem Case).

Bei `cards+` gibt es ein einfaches Kriterium, das beim Aussortieren von Information hilft. Eine Information, für die es einen passenden Baustein in `cards+` gibt, wird dauerhaft erfasst und inkrementell gepflegt. Alle andere Informationen, die zu einem bestimmten Zeitpunkt zwar wichtig und notwendig sind, später aber nicht mehr gebraucht werden, gehören zur Projektdokumentation. Dazu zählen beispielsweise User-Stories in einem Backlog, Studien oder Analysen.

Mit `cards+` werden Entwickler zu Autoren, ohne sich extra anstrengen zu müssen. Sie nutzen Formate wie `Asciidoc` für Spezifikationen, die sich wie Code anfühlen. Sie nutzen Formate wie `JSON`, `XML` oder `YAML` für Konfigurationen, die durch Kommentare lesbar sind. Sie generieren Dokumentation direkt aus dem kommentierten Code. `cards+` als Werkzeug macht dokumentierte Informationen im Quelltext und in Testplänen für die Dokumentation im Wiki nutzbar.

Wird das Konzept `cards+` von Anfang an in der Projektorganisation gelebt, dann ist eine hochwertige Produktdokumentation keine Illusion mehr. Der Aufwand für die Herstellung der Dokumentation mit den Prinzipien und Praktiken von `cards+` unterscheidet sich nicht von dem Aufwand, Dokumente "irgendwie" zu schreiben. Aber die Chancen, die sich durch eine strukturierte Produktdokumentation in einem Wiki ergeben, sind nur durch die Kreativität der Personen beschränkt, die sie lesen und pflegen.