10 Tipps, damit Ihr Projekt auch sicher schief geht

((5.777 Zeichen))

Sabine Link, Method Park

Tester & Entwickler: good guy vs. bad guy?

Jeder, der schon einmal in Software-Entwicklungsprojekten gearbeitet hat, kennt das Problem: Entwickler, die Software erstellen, und Tester, die auf Funktionalität prüfen und Qualität sichern, sind irgendwie unterschiedliche Typen.

Das beginnt bereits bei der grundsätzlichen Herangehensweise: Die einen (Entwickler) sind eher konstruktiv und von Haus aus Optimisten, sie erstellen den Programmcode, sind von sich und der Funktionalität ihrer Software restlos überzeugt. Die anderen (Tester) sind eher destruktiv und berufsmäßige Pessimisten, sie finden immer ein Haar in der Suppe und sind darauf aus Fehler zu entdecken.

Kein Wunder, dass es zwischen den verschiedenen Lagern oft Schwierigkeiten und fehlende Wertschätzung gibt. Entweder reden Entwickler und Tester gar nicht erst *mit*einander oder sie reden *an*einander vorbei. Besonders schlecht ist es, wenn beide *gegen*einander arbeiten und sich gegenseitig als natürliche Feinde betrachten.

Dabei ist eine gelungene Zusammenarbeit dieser Rollen, die sich respektieren und gut in ihren Fähigkeiten ergänzen, entscheidend für den späteren Projekterfolg.

Die folgenden nicht ganz ernst gemeinten Vorschläge wollen deshalb auf einige typische Problemsituationen und Stolperfallen im Projektalltag aufmerksam machen.

Was Tester und Entwickler tun können, um gute Zusammenarbeit zu verhindern

Stellen Sie sich vor, Sie sind in einem Software-Entwicklungsprojekt und es gibt keinerlei Probleme, alles klappt wie am Schnürchen. Der Freigabetermin wird gehalten, Tester und Entwickler verstehen sich und arbeiten perfekt zusammen. Das wäre doch langweilig, oder?

Hier sind zehn hilfreiche Tipps, damit das nicht passiert, sondern etwas Schwung in die Sache kommt, so dass es im Projekt wieder spannend wird.

1. Grundsätzlich sollten Entwickler und Tester nie miteinander reden. Das ist ganz schlecht und kostet nur unnötig Zeit. Am Ende könnte man sich ja noch verstehen und Unklarheiten beseitigen. Kommunikation ist eh überbewertet. Schließlich sind wir Informatiker und keine Sozialpädagogen!

2. Entwickler und Tester leben in verschiedenen Welten, und das ist auch gut so. Deshalb sollten sie sich auch nicht gegenseitig helfen oder gar zusammenarbeiten. Alleine Probleme lösen dauert viel länger, da ist man wenigstens eine Weile beschäftigt. Am besten wirft man die Software ohne Kommentar über den Zaun und gibt keine Tipps zu etwaig bekannten Problemen.

3. Wenn es sich dann doch nicht vermeiden lässt und beide Seiten miteinander reden müssen, dann sollte man am besten möglichst viele Fachbegriffe verwenden, die der andere nicht versteht. So kann man effektiv viel Sachkenntnis vortäuschen. Das wirkt kompetent und verhindert lästiges Nachfragen. Wenn schon kommunizieren, dann auch bitte in langen, unnötigen Meetings, bei denen man möglichst viel Zeit verbraucht und alle einschläfert.

4. Wenn der Tester einen Fehler findet, dann kann der Entwickler einfach sagen, dass das ein Benutzer ohnehin so nie machen würde. Das Testszenario sei außerdem völlig realitätsfremd. Am besten ist es, wenn man jeden Fehler ausdiskutiert und der Entwickler dem Tester ausredet, dass da ein Problem sein könnte. Normalerweise ist das Ganze ein Feature und kein Bug, der Tester sieht das nur nicht ein.

5. Entwicklertests sind etwas für Angsthasen. Schließlich sagt das Wort schon, dass das eine Testaufgabe ist. Qualitätssicherung ist alleinige Tätigkeit der Testabteilung. Und als Entwickler sollte man bloß keine Spezifikationen lesen. Man weiß, was für den Anwender am besten ist; und falls die Programmierung von den Anforderungen abweicht, dann wird der Test das schon finden, dazu ist er ja da.

6. Auch Regressionstests sind nicht nötig. Der Entwickler kennt schließlich seine Software und hat alles im Griff. Er weiß genau, an welcher Stelle eine Software-Änderung Seiteneffekte haben kann und wo definitiv nicht. Man kann es sich also sparen, bereits erfolgreich abgeschlossene Tests zu wiederholen, nur um sicherzustellen, dass sie noch funktionieren. Es ist doch klar, dass alles passt.

7. Wenn der Tester die schlechte Nachricht einer Fehlermeldung überbringt, dann sollte er den Entwickler am besten gleich persönlich beleidigen, denn der ist ja schließlich Schuld, dass nichts mehr geht und man nicht weiterarbeiten kann. Das hätte man selbst sowieso viel besser gekonnt und solche Anfängerfehler wären nie passiert.

8. Tester müssen nicht genau wissen, was sie gemacht haben, als der Fehler auftrat. Der Entwickler findet das dann schon raus. Schuld ist der sowieso. Es ist auch nicht so wichtig zu wissen, auf welcher Konfiguration getestet wurde. Fehler ist Fehler und es ist total unwahrscheinlich, dass das Problem am Testskript oder der Installation liegt.

9. Gegenseitiger Respekt ist nicht angebracht. Ein Tester ist jemand, bei dem es nicht zum Entwickler gereicht hat. Und das bisschen Testen kann ja schließlich nicht so schwer sein. Auf der anderen Seite hat der Entwickler sowieso keine Ahnung, wozu die Software da ist und wie sie benutzt wird. Es wäre das erste Mal, dass der technikverliebte Programmierer Überblick über das Gesamtsystem hätte.

10. Termine und Fristen sind dazu da, dass man sie überzieht, da muss man auch nicht miteinander kommunizieren oder Zeitpläne anpassen. Pufferzeiten braucht kein Mensch. Man kann schon mal etwas länger programmieren, das ist schließlich die entscheidende Tätigkeit bei der Software-Entwicklung. Der Tester schafft es auch kurz vor Release noch, alles zu testen und spontane Änderungen

zu berücksichtigen. Dann muss er halt mal ein paar Nachtschichten einlegen, das geht schon. Und notfalls testet halt der Anwender.

Mit diesen einfachen Maßnahmen sind Probleme im Projekt garantiert. Halten Sie sich daran und es wird schon schiefgehen!

Autorin

Lebenslauf Sabine Link

Sabine Link studierte Informatik/Wirtschaftsinformatik an der Friedrich-Alexander-Universität in Erlangen-Nürnberg und hat den Master of Computer Science der FernUniversität Hagen. Seit 2001 ist sie Mitarbeiterin der Method Park Software AG in Erlangen und dort in den Bereichen Systemtest, Prozessverbesserung, Coaching und Testmanagement für verschiedene Kundenprojekte tätig. Außerdem ist Sabine Link Trainerin für Seminare zum Thema Software-Test und Autorin zahlreicher Fachartikel in Zeitschriften sowie des Buches „Verbesserung von Softwaretestprozessen – Ein Prozess-Assessment-Modell für TPI nach SPICE".

Kontakt

Method Park Holding AG
Wetterkreuz 19a
91058 Erlangen
www.methodpark.de
Sabine.Link@methodpark.de
Tel. +49 9131 97206-319
Fax +49 9131 97206-200