Martin Klonk *04. April 2017*

Das perfekte Testkonzept – mit Leichtigkeit!



© pathdoc / Fotolia.com

Der kleine Prinz braucht das Schaf – und nicht der Erzähler...

Der Ruf nach einem Testkonzept klingt zunächst einmal für die Betroffenen befremdlich bis beängstigend. Zu Recht! Aber nur auf den ersten Blick...

Denn mit wenigen Schritten ist das Problem schneller gelöst als gedacht. Und es ist immer sinnvoll, sich die Dinge im Vorhinein gut zu überlegen. Erst Recht in agilen Projekten. Aber was genau ist ein Testkonzept und wozu braucht man es? Zunächst wollen wir einmal mit den zumeist antiquierten Vorstellungen darüber aufräumen. Anschließend entwickeln wir Schritt für Schritt, wie man sicher zu einer erfolgreichen Testkonzeption gelangt. Ach ja, das Testkonzept entsteht damit ganz von allein.

Der Grundgedanke: Was soll ein Testkonzept leisten?

"*Bitte … zeichne mir ein Schaf!*" ruft der kleine Prinz im gleichnamigen Kinderbestseller dem Erzähler zu, der zunächst verblüfft und dann leider untalentiert mehrere Versuche unternimmt, ein Schaf zu malen [1]. Erfolglos! So wie dem Erzähler, der eigentlich dabei ist, ein Flugzeug in der Wüste zu reparieren und plötzlich ein Schaf malen soll, so geht es vielen Projektmitarbeitern, die plötzlich ein Testkonzept erstellen sollen. Leider entstehen dann – ebenso untalentiert – Konvolute, die niemanden wirklich weiterhelfen und die vor allem außer dem Autor kaum noch jemand später in die Hand nimmt. Mit dieser frustrierenden Erfahrung nimmt dann das landläufige Missverständnis seinen Lauf, Testkonzepte seien überflüssige Reste einer längst veralteten Kultur in der Softwareentwicklung.

Eine überzeugend gezeichnete Box mit Luftlöchern – das Schaf braucht schließlich frische Luft – löst dann das Problem des Erzählers mitten in der Wüste. Den Rest besorgt die Fantasie des kleinen Prinzen und endlich ist der "Kunde" zufrieden. Letztendlich brauchte der kleine Prinz das Schaf – und nicht der Erzähler. So verhält es sich auch mit der Forderung eines Testkonzepts. Die naturgetreue Darstellung des Tests ist meist unzulänglich. Wer weiß denn wirklich, was kommen wird! Meist verzettelt man sich auch nur. Aber einen Rahmen für das, was da drin sein soll, kann man immer zeichnen. Den Rest füllen die Teamkollegen dann – vielleicht nicht mit Fantasie – aber mit Sachverstand und Gegenwärtigkeit.

Der Rahmen: Die Norm IEEE 829 als Checkliste zur Klärung der Fragen

Ein guter Rahmen für das Testkonzept (engl. test plan) ist in der Norm IEEE 829-2008 [2] hinterlegt. Diese Norm ist auf den ersten Blick bürokratisch und wirklich nicht sehr modern, gerade aber in Bezug auf das Testkonzept sehr nützlich, weil sie uns wie eine Checkliste durch die Fragen führt, die wir in der Testkonzeption für uns (und andere) beantworten müssen.

Die Norm unterscheidet zwischen Master- und Stufen-Testkonzept, was zu unterscheiden aber nur in großen Projektumfeldern sinnvoll ist. Darauf kommen wir später noch einmal kurz zurück. Grundsätzlich gliedert sich ein Testkonzept nach dieser Norm in folgende Abschnitte:

IEEE 829	Deutsche Übersetzung (Erläuterung)
Test plan identifier	Testkonzept-ID
Introduction	Einführung (Zusammenfassung)

Autor



Martin Klonk

Martin Klonk ist Senior Berater für Softwaretests, Mitglied des Austrian Testing Boards und Co-Autor des Buches "Agile Testing". Er hat an den Lehrplänen des...

Publikationen

Agile Testing: Der agile Weg zur Qualität



Newsletter

Unser Newsletter informiert Sie regelmäßig und kostenlos über Neuigkeiten, Artikel und Veranstaltungen zu aktuellen IT-Themen.

Newsletter abonnieren

Aktuelles

04.11.2025

DevSecOps mit GitLab: Kostenfreie Roadshow in

IEEE 829	Deutsche Übersetzung (Erläuterung)
Test items	Testgegenstand
Features to be tested	Zu testende Eigenschaften
Features not to be tested	Nicht zu testende Eigenschaften
Approach	Testansatz (die gewählten Teststrategien)
Item pass/fail criteria	Kriterien für Testende und -abbruch
Suspension criteria and resumption requirements	Kriterien für Testunterbrechung und -wiederanlauf
Test deliverables	Testergebnisse (Arbeitsergebnisse)
Testing tasks	Testaktivitäten
Environmental needs	Erforderliche Voraussetzungen (u.a. Testumgebungen)
Responsibilities	Zuständigkeiten
Staffing and training needs	Besetzung und Trainingsbedarf
Schedule	Testplan
Risks and contingencies	Risiken und Notfallplan
Approvals	Genehmigungen / Freigabe

Viele Testkonzepte orientieren sich in ihrer Gliederung an diesem Gerüst. Aber erstens hält das niemand 1:1 durch und zweitens muss es ja nicht wirklich ein Dokument sein. Gerade agile Teams werden eher ein Set an Regeln und Praktiken brauchen, auf die man sich einigt. Diese manifestieren sich eher in Projekt-Wikis oder – wie bei Kanban eher üblich – explizit formulierte Regeln neben dem Taskboard.

Das Vorgehen: Eher im Dialog Dinge durchdenken und dann vereinbaren

Auch müssen wir dringend mit der Vorstellung aufräumen, ein Testmanager säße wie ein Handwerker – oder treffender – ein musikalisches Genie im stillen Kämmerlein und entwerfe da sein großes Werk. Nein, ein Testkonzept entsteht schrittweise im Dialog mit Experten und zukünftig Betroffenen im Projekt. Wie wir gleich noch sehen werden, reicht bei kleinen Vorhaben möglicherweise schon ein halber bis ein ganzer Tag (fairer Weise sei hier erwähnt, dass eine Testkonzeption i. d. R. zwischen 5 bis 20 PT Aufwand beansprucht – bei ganz großen Vorhaben (Mulitprojekte, Programme) auch deutlich mehr) mit den richtigen Experten, um all die nötigen Fragen zu klären und ein paar Flipcharts mit den Ergebnissen, um die Testkonzeption durchzuführen. Sicher sind im Anschluss noch einzelne Details auszuarbeiten, aber das wichtigste sollte angerissen und einmal gemeinsam durchdacht sein. Alternativ kann ein Testmanager, Testkoordinator, Testverantwortlicher (oder wie auch immer man ihn nennen mag) in Einzelinterviews die Fragen klären und alles nach und nach zusammentragen.

In agilen Teams, in denen iterativ gearbeitet wird, müssen nicht alle Fragen auch vorweg beantwortet sein. Es reicht auch, sie anzureißen und ihre Lösungen nach Prioritäten der Reihe nach anzugehen. So ist nicht einzusehen, warum ein Team von Anfang an genau definieren muss, welche Testartefakte (Testfälle, Testprotokolle, Berichte, ...) es führen will. Wenn aber ein Akzeptanztest durch den Fachbereich zum nächsten Release erforderlich ist, sollte man die Zielgruppe schon jetzt lieber buchen (und sich damit unweigerlich festlegen). Die o. g. Norm in Form einer Taskliste könnte aber die nötigen Klärungen gut visualisieren und dafür sorgen, dass man im Laufe der Iterationen nichts zu spät löst bzw. entscheidet.

Wichtig bei allen Ausarbeitungen und Festlegungen: Man braucht die unbedingte Zustimmung der Beteiligten! Ein Review-Zyklus dient also nicht der Fehlerfindung im Konzept allein, sondern vor allem der Zustimmung. Diese muss man sich aktiv einholen. Viele machen hier den Fehler, ein Konzept auf den Tisch zu knallen und dann davon auszugehen, dass sich alle (gefälligst) daran halten. Systemisch liegt dann die Verantwortung bei dem Ersteller des Testkonzepts und er wird sich um alles selbst kümmern müssen, was selten sinnvoll ist.

Schritt 1, Arbeitsteilung im Test: Teststufen oder Testquadranten geben Orientierung

Es gibt nicht *den* Test! Es gibt verschiedenste Tests, die aus verschiedensten Perspektiven das zu testende System unter die Lupe nehmen. Und das muss auch so sein.

Der eine Aspekt ist die Integrationsstufe. Hier hilft Testern vor allem das V-Modell mit seinen 4-5 Integrationsstufen, um zu verstehen, dass Tests auf jeder Integrationsstufe etwas anders

Frankfurt, Zürich, München, Düsseldorf, Berlin und Wien
Neuigkeiten zu GitLab gibt es kostenfrei bei Snacks und Getränken bei der Roadshow in Frankfurt,...

25.10.2025

Europäische Alternativen zu Cloud-Diensten
Digitale Souveränität: Welche europäischen Cloud-Dienste gibt es, die digitale Souveränität bieten?...

09.09.2025

Konferenz für Java-Entwicklung 2025: Java-Tage in Frankfurt am Main
Die Java-Tage finden vom 8. bis 11. Dezember 2025 in Frankfurt/Main statt. Sie sind in die IT-Tage,...

IT-Jobs auf Informatik Aktuell

vorgehen. Das ISTQB [3] teilt die Testaufgaben daher in folgende Stufen ein:

- *Reviews* auf Anforderungen, Design, sonst. Spezifikationen – ist schließlich auch eine Form des Testens!
- *Komponententest* – Entwickler testen ihre einzelnen Komponenten
- *(Komponenten-)Integrationstest* – Das Zusammenspiel der Komponenten wird ggf. schrittweise (wie z. B. bei Continuous Integration) getestet
- *Systemtest* – Das zu liefernde System wird als Ganzes getestet (ist damit eher der Warenausgangstest der IT)
- *Abnahmetest* – Das gelieferte System wird durch Abnehmer bzw. Nutzer auf Anwendbarkeit/Nutzbarkeit geprüft. Daher oft auch Akzeptanztest genannt
- *(System-)Integrationstest* – Das System muss sich im Zusammenspiel in der gesamten Anwendungslandschaft bewähren. Manche Geschäftsprozesse oder Lastszenarien sind erst hier wirklich abbildbar.

Für agile Teams liegen diese Stufen zu nahe beieinander (sowohl zeitlich, als auch personell). Daher ergibt eine Trennung in verschiedene Stufentestkonzepte (wie in der oben zitierten Norm IEEE 829 vorgesehen) hier nur wenig Sinn – und es stellt sich die Frage, ob die vier Testquadranten nicht hilfreicher sind, aber dazu noch weiter unten. Bei anderen Projekten wiederum entziehen sich die Arbeiten des Lieferanten des Zuständigkeitsbereiches der Tester, weshalb man sich auf Abnahmetests und Systemintegrationstests konzentriert. Ansonsten kann man überlegen, ob man alles erst einmal in einem Master-Testkonzept konzipiert und dann den jeweiligen Teams für die Teststufen noch auferlegt, ein konkretisierendes Stufentestkonzept auszuarbeiten. Lohnt sich aber nur in stark arbeitsteilig organisierten klassischen Projekten.

Für agile Teams hat sich aber inzwischen ein Modell durchgesetzt, dass die verschiedenen Testperspektiven etwas vereinfachend, aber umso wirkungsvoller darstellt. Die sog. vier Testquadranten von Crispin & Gregory [4] teilen die Testaufgabe in eine eher IT-technische und eine eher fachliche Perspektive bei den Tests. Außerdem werden Tests unterschieden, je nachdem, was man im Vorhinein schon ausgearbeitet hat (*supporting the team* – Team unterstützend) und auf welche Probleme man erst im Laufe der Fertigstellung stößt (*critique the product* – Produkt hinterfragend) .

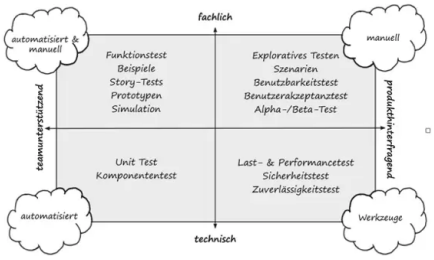


Abb.1: Die vier Testquadranten von Crispin/Gregory. Quelle: Agile Testing

Damit lassen sich ebenfalls verschiedene Testarten (z. B. Unit-Test, Benutzbarkeitstest) unterteilen, für die es ggf. verschiedene Mitarbeitergruppen gibt, die sich jeweils darum kümmern. Gerade bei Last-/Performancetests, Sicherheitstest und Zuverlässigkeitstests sind oft Hilfestellungen von Partnern außerhalb des Projektteams nötig.

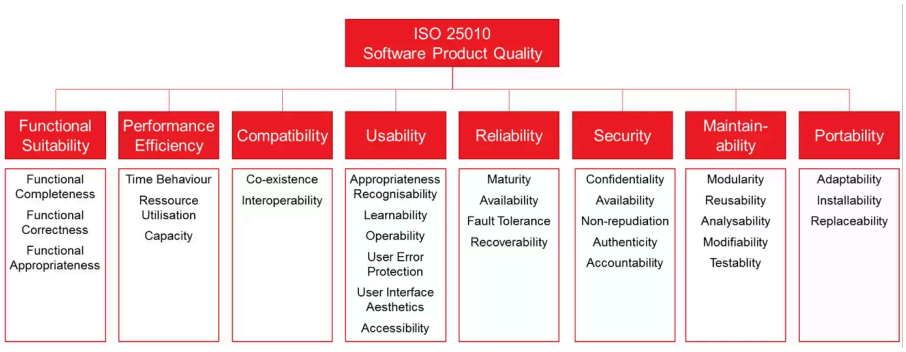
Vielleicht wird die Eine oder der Andere jetzt fragen, wo der Regressionstest bleibt. Dieser ist kein eigener Test! Er ist immer Teil eines Tests und sollte bei jedem Test mitgedacht werden, auch wenn man ihn evtl. organisatorisch (z. B. Nutzung von Offshore-Mitarbeitern) oder technisch (durch automatisierte Tests) auslagert.

Schritt 2, Was testen und was nicht: Qualitätsmodell ISO 25010 als Ideengeber

Wenn man die Perspektiven des Tests und damit ggf. auch die Arbeitsteilung im Test geklärt hat, sollte man sich vor Augen führen, was überhaupt der Testgegenstand ist.

Zum einen wären natürlich die Funktionen/Features (oder wie sie jeweils im Projekt genannt werden) zu nennen. Hier ist genau zu unterscheiden, was zu testen ist und was genau nicht. Bitte wirklich beides überlegen! Manchmal ist eine Funktion besser dadurch abzugrenzen, was nicht zum Funktionsumfang gehört. Bei einem Webshop kann man beispielsweise gerne die Abbruchfunktion für das Einkaufen versprechen, aber es macht die Sache doch für den Auftraggeber klarer, wenn man einschränkt, dass der Kunde dann (noch) nicht in der Lage ist, mit dem selben Warenkorb erneut wieder zur Kasse zu gehen.

Zum anderen sollte man aber auch quer zu den Features auf die nicht funktionalen Qualitätseigenschaften schauen. Aktuell haben wir einen Trend, bei dem Zuverlässigkeit und Sicherheit bei IT-Projekten absolut im Vordergrund stehen. Aber auch Benutzbarkeit ist ein häufig unterschätztes Thema. Hier empfiehlt sich ein Blick auf das Qualitätsmodell der ISO 25010 [5], das man als Checkliste der Reihe nach durchgehen sollte.



Nun ist es wichtig, dieses Modell nicht 1:1 in sein eigenes Testkonzept zu übernehmen, sondern sich anhand der Kriterien zu fragen: Ist das wirklich wichtig für unser Projekt? Wenn ja, können wir das gezielt prüfen? Haben wir die Kompetenz dazu?

Daraus ergibt sich, ob noch bestimmte nicht-funktionale Testperspektiven fehlen, für die man ggf. mit Hilfe Dritter (zentrale Experten eines Testcenters, externe Firmen, anerkannte Experten in anderen Projekten) arbeiten sollte.

Schritt 3, Das Richtige tun: Prioritäten setzen und Strategien entwickeln

Nun ist man meist an dem Punkt, wo man fast schon erstarrt vor der Last, die da scheinbar an Testnotwendigkeiten auf einen zukommt. In Wirklichkeit ist nicht alles gleich wichtig und relevant. Um ein Beispiel zu geben: Ein Web-Portal, das Anmeldungen (für Sportkurse, Förderprogramme) nach dem Prinzip first-come, first-served zu vergeben hat, kämpft eher mit Sicherheit bzw. Last/Performance. Die Funktionalität ist dabei äußerst simpel. Dementsprechend muss man auch Mut zur Lücke zeigen, Unwichtiges auf die Kernfunktionen beschränken und den Rest möglicherweise bis auf ein paar Stichprobentests reduzieren. Man nennt dies eine risikoorientierte Teststrategie.

Es gibt aber auch andere Teststrategien, über die man sich ergänzend Gedanken machen sollte. So z. B. die Strategie, möglichst früh zu testen. Je früher Fehler gefunden werden, desto (deutlich) geringer sind die Kosten für die Behebung. Beispielsweise kann ein agiles Team zwar hausinterne Experten mit aufwändigen Performancetests gegen Ende eines Releases beauftragen. Das könnte dann aber schon zu spät sein, um noch adäquat zu reagieren. Warum nicht schon im Vorfeld Annahmen über die neuralgischen Schwächen treffen und diese mit selbstgeschriebenen (günstigeren) JMeter-Tests fast täglich gezielt durchmessen? So macht man Verbesserungstrends sichtbar oder kann bei plötzlichen Verschlechterungen sofort Alarm schlagen. Die o. g. aufwändigen Performancetests sollte man dann zur Bestätigung trotzdem durchführen lassen.

Bei dieser Gelegenheit kann man eher einmal auf den Regressionstest zurückkommen und sich fragen, mit welcher Strategie man diese Aufgabe bewältigt? Alles zur Sicherheit immer wieder testen, weil funktionierende Software nach einer Änderung und Erweiterung plötzlich nicht mehr funktionieren könnte? Das macht keiner! Aber nur die neuen Features zu testen, ist in den meisten Fällen grob fahrlässig. Wer früh auf Testautomatisierung, möglichst noch auf Unit-, aber zumindest auf Service-Ebene setzt (s. das Konzept der Testpyramide von Mike Cohn [6]), kann sich sicher ein größeres Testportfolio leisten, als jemand, der ganz auf manuelle Tests angewiesen ist. Dr. Rößler hat hierzu kürzlich einen ergänzenden Ansatz in Richtung KI vorgestellt , was evtl. für Sie auch eine sehr interessante Option sein könnte [7] .

Schritt 4, Testumgebung und Testdaten: Rechtzeitig die Infrastruktur beantragen

Und nun einmal eines ganz fest hinter die Ohren schreiben: Testdaten und Testumgebung brauchen in den meisten Fällen besonders langfristige Vorbereitung und viel Zuarbeit von Mitarbeitern außerhalb des Projektteams, die sicher nicht bereitwillig auf Aufträge oder Amtshilfegesuche von Ihnen warten.

Daher sollte man früh die Frage klären, wie produktivnah die Umgebung sein muss. Kann man ggf. hier auf Cloudlösungen zurückgreifen [8] ? Bei nicht-funktionalen Tests, wie Sicherheit- und Last-/Performance- ist das ein sehr unangenehmes Thema. Stimmen Sie sich hier mit Experten einmal ab. Vielleicht kann man die Testumgebung so geschickt auf die zu erwartenden Schwachstellen zusammenstutzen und sich so eine zweite Produktivumgebung ersparen.

Auch bei den Testdaten wird es heikel. Die Verwendung von Produktivdaten ist in vielen Fällen sogar datenschutzrechtlich verboten. Und laut EU Daten-Schutz-Grundverordnung wird es ab 2018 in den EU-Staaten noch strenger [9] . Solange nur ein Fachbereich als Tester auftritt, tut man sich hier leichter. Aber für alle anderen Tester sind synthetisch erzeugte Testdaten bzw. anonymisierte Testdaten aus Produktion ein Muss. Es gibt über die Werkzeuge

für Testautomatisierung inzwischen auch hier gute Methoden, sich einen qualitativ wertvollen Testdatenbestand selbst zu erzeugen.

Schritt 5, Testaktivitäten: Zusammenarbeit und Zuständigkeiten grob skizzieren

Nun endlich kommen wir zu dem klassischen Testplan: Dem "Wer macht Was?". Solange man noch in Gesprächen mit Anderen oder in der Gruppe ist, sollte man die Zuständigkeiten festlegen. Wer ist für was der Richtige – und könnte er das auch übernehmen? Braucht man noch zusätzliche Ressourcen oder Hilfe von Experten außerhalb des Projektteams (z. B. für Sicherheit, Last/Performance)?

Anschließend sollte man eine vorsichtige Schätzung machen. In agilen Teams reicht es, die nächste Iteration zu schätzen und dann von Iteration zu Iteration die Schätzmethoden zu verfeinern. In klassischen Projekten sollte man zunächst wirklich alle Aktivitäten auflisten, die vermutlich anfallen werden. Ggf. schon einmal Arbeitseinheiten wie Testobjekte oder Features etc. abgrenzen. Danach sich mit Experten oder anderen erfahrenen Testern im Vier-Augen-Prinzip über die Schätzung beugen und alles im Kopf einmal durchspielen. Dieser schnelle Top-down/Bottom-up-Ansatz bewährt sich immer, sofern man es nicht mit der Genauigkeit übertreibt und letztendlich akzeptiert, dass man vieles noch nicht genau weiß.

Schritt 6, Risiken und Notfall: Von Geheimdiensten lernen

Der Klassiker unter den Projektverfehlungen ist, wenn der Testmanager von einer verspäteten Lieferung überrascht wird und dann nicht mehr rechtzeitig das Ruder herumreißen kann. Dabei gehört die verspätete oder unvollständige Lieferung zu einer Standardsituation! Warum gibt es also dafür keinen Plan B?

Geheimdienste, so habe ich mir zumindest sagen lassen, haben für jede Operation einen sog. Plan B. Also eine Schadensbegrenzung, wenn die Sache übel auffliegt. Kann man seine Agenten dann immer noch in Deckung bringen oder retten? Wer opfert sich ggf. freiwillig, welcher Politiker hat schon die richtige diplomatische Avance in der Schublade, um Schlimmeres zu verhindern?

Genau das sollte man sich für den Test auch schon zurechtlegen. Ein klassisches Brainstorming unter Experten: Was kann alles schiefgehen? Wie könnten wir uns dann aus der Schlinge ziehen? Wenn sich z. B. die Tests als komplizierter herausstellen als gedacht, habe ich dann mit dem Abnehmer bzw. Kunden schon vereinbart, welche Teile wir ggf. opfern können, um dennoch pünktlich produktiv zu gehen? Je länger der Angesprochene Zeit hat, sich das fair zu überlegen, desto treffsicherer ist seine Wahl.

Zu diesem Spiel gehört aber ganz klar gegenseitiges Vertrauen – zumindest Geheimdiplomatie zwischen Lieferant und Kunde. Ich muss als Tester/Testmanager glaubhaft sein darin, dass ich den Joker wirklich nur dann ziehe, wenn das Unvorhergesehene zuschlägt. Das ist kein Freibrief für Mittelmäßigkeit in der Leistung!

Ausblick: Warum immer das Rad neu erfinden?

Nun, am Ende angekommen, sieht man, dass da doch sehr viel zu ermitteln, planen und zu entscheiden ist. Ich kann nur immer wieder betonen, dass diese 6 Schritte sich in einem Workshop zügig durchsprechen lassen, solange man sie als Roadmap im Blick behält. Und sicher ist dann noch eine Menge an Detailarbeit – vor allem in klassischen Projekten – nötig. Bei agilen Teams verteilt sich die Arbeit dann peu à peu auf die Iterationen.

Viele dieser Fragen stellt man sich möglicherweise immer wieder und löst sie möglicherweise (zumindest innerhalb einer Organisation) auf die gleiche Art und Weise. Es ist daher nur anzuraten, diese Lösungen in die Gemeinschaft der Tester bzw. Projekte zu tragen und als Best-Practices allen verfügbar zu machen. So etwas nennt sich dann Testhandbuch oder – wenn es weniger Papierkram sein soll – eine *Community of Practice*.

Noch einen Schritt weitergehend kann man über eine eigene Excellence-Organisation für Test verfügen, die – nicht so bürokratisch wie die verhassten Testcenter-Behörden – in einer Organisation Test-Expertise für die Projekte verfügbar machen. So könnte beispielsweise ein Testcoach einem Testverantwortlichen am Anfang des Projektes bei den 6 oben skizzierten Schritten helfen und so die Best-Practices in der Organisation auch diesem Team verfügbar machen. Denken Sie einmal darüber nach! Auch bei der Bereitstellung von Testumgebungen, Testdaten oder Werkzeugen könnte das ein interessanter zentraler Service sein.

Fazit

Gehen Sie nicht blind in ein Testvorhaben hinein. Man muss zwar nicht den Irrtum durch den Plan ersetzen, aber es hilft, sich einmal alles in 6 Schritten möglichst gemeinsam zu durchdenken. Ob dabei wirklich ein Dokument entsteht, ist zweitrangig. Wichtig ist, dass alle Betroffenen wissen, wann sie am Zug sind und was von Ihnen abhängt.

Gut ist, wenn es in der Organisation übergreifende Testservices gibt, die mit Experten/Coaches die Projektteams bei der Ausarbeitung unterstützen können oder

nachhaltig für einen Erfahrungsaustausch zwischen Testern sorgen.

Quellen

[1] Antoine de Saint-Exupéry, Der Kleine Prinz, Karl Rauch Verlag, Düsseldorf, 1956
[2] IEEE Std 8292008, IEEE Standard for Software and System Test Documentation, The Institute of Electrical and Electronics Engineers, Inc., New York, NY 100172394, USA, 2008
[3] ISTQB – International Software Testing Qualifications Board, Certified Tester Foundation Level Syllabus, 2011
[4] Crispin, L., Gregory, J., Agile Testing – A practical Guide for Testers and agile Teams, AddisonWesleyLongman, Amsterdam, 2009
[5] ISO/IEC 25010:2011, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models, 2011
[6] Cohn, M., The Forgotten Layer of the Test Automation Pyramid, 2009
[7] Informatik Aktuell – Rößler, Dr. J.: 5 Dinge, die Sie über Regressionstests wissen sollten
[8] Informatik Aktuell – Grötz, R. / Hofmann, B.: DevOps – Testautomation II – Webapplikationen & Mobile Device Clouds
[9] Bruckner, Regina, Die Leichtigkeit des Datensammelns ist bald passé, Der Standard, Aug. 2016

Autor



Martin Klonk
Martin Klonk ist Senior Berater für Softwaretests, Mitglied des Austrian Testing Boards und Co-Autor des Buches "Agile Testing". Er hat an den Lehrplänen des...

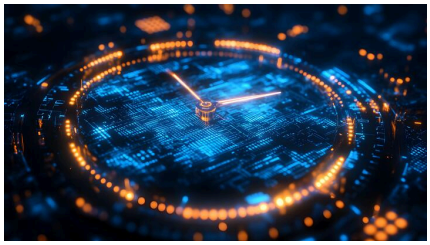
Publikationen

- Agile Testing: Der agile Weg zur Qualität

Das könnte Sie auch interessieren



Die Ära von "Buy vs. Build" ist vorbei
Tino Fliege



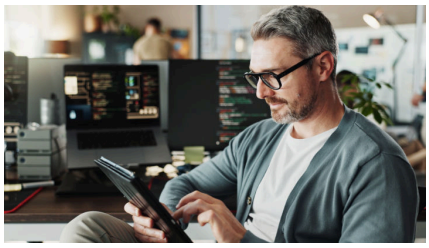
Barrierefreiheit automatisiert testen
Anna Maier



Die Architektin als Gestalterin von Strukturen
Frank Müller



Digitale Barrierefreiheit 2025 – Was IT-Teams jetzt technisch umsetzen müssen
David Dutschke



WinUI 3 – State of the Art
Elena Bochkor & Dr. Veikko Krypczyk



Mit User Experience zur besseren Software-Architektur
Sarah Ganter

Kommentare (0)

Neuen Kommentar schreiben

Name:

E-Mail-Adresse:

Kommentar:

☐ Ich verstehe und akzeptiere die Datenschutzbestimmungen.

Absenden

© Alkmene Verlag GmbH 2025