ca technologies

Der ultimative Leitfaden für kontinuierliches Testing

2018

So wird Ihr Weg zum kontinuierlichen Testing erfolgreich

KAPITEL 1

Das Tempo für Unternehmen steigt stetig. Sie beginnen daher, zu erkennen, dass sie ihren Entwicklungs- und Releaseprozess für Software verändern müssen, um wettbewerbsfähig zu bleiben. Releases werden heute weit häufiger veröffentlicht. Es ist nicht mehr möglich, Fehler in Zyklen von sechs bis 18 Monaten zu finden und zu beheben, ohne die Kunden zu verärgern. Alles muss schneller gehen und schneller bereit und perfekt sein.

Es findet eine tief greifende Veränderung statt. Die Verantwortung für das Testing beginnt früher im Softwareentwicklungszyklus („Shift Left") und erstreckt sich über den gesamten Zyklus. Damit schwindet die herkömmliche Trennung zwischen Entwicklern und Testern, und alle sind nun für die Qualität verantwortlich. Das Testing wird wichtiger, da die Konsequenzen unsachgemäßen Testing besser sichtbar werden.

Dies ist eine technische Veränderung, die wiederum eine Veränderung der Unternehmenskultur erfordert. Wenn sich das Testing weiterentwickelt und auf den gesamten Softwareentwicklungszyklus (SDLC) ausdehnt, müssen Unternehmen und ihre IT-Abteilungen eine Möglichkeit finden, die gesamte Unternehmenskultur im Bereich Softwareentwicklung und -Testing zu verändern.

Vielleicht ganz einfach so: Wir müssen aufhören, uns das Testing als ein Ereignis vorzustellen. Es ist nicht etwas, was zu einem bestimmten Zeitpunkt stattfindet. Stattdessen müssen alle es ständig tun, sogar vor Beginn der Entwicklung. Fehler und Probleme müssen sofort behoben werden, nicht irgendwann erkannt oder gar übersehen werden, bis das Produkt beim Kunden landet.

Bei DevOps geht es darum, diese Trennung zwischen Entwicklung und Betrieb zu überwinden, um dafür zu sorgen, dass Software (die hoffentlich mit der Agile-Methode erstellt wurde) schnell für die Anwender bereitgestellt werden kann, sobald sie dafür bereit ist, ohne dass die Qualität leidet. Auch Unternehmen, die nach der DevOps-Methode arbeiten, gehen weiterhin Kompromisse zwischen Qualität und Geschwindigkeit ein.

Mit der aktuellen Herangehensweise an das Testing treten 63 % der Verzögerungen bei der Softwareentwicklung durch Testing und QA im Lebenszyklus auf, und 70 % des Testing werden noch manuell durchgeführt. Dies ist jedoch nicht mehr praktikabel. Sie können Tests nicht einfach manuell durchführen. Außerdem sollten Sie weitere Probleme bedenken:

56 % der kritischen Abhängigkeiten sind nicht verfügbar.

50 % der Zeit werden mit der Suche nach Testdaten verbracht.

64 % der Fehler treten in der Anforderungsphase auf.

Fehler – vor allem in Form von Ausfällen, die die Anwender betreffen – schaden Ihren Daten, Ihren Business-Prozessen, der Kundenakzeptanz, der Kundenbindung und Ihren Marken. Mit veralteten Wasserfallmodell- oder sogar Agile-Prozessen können Sie keine hohe Geschwindigkeit und Qualität für Ihren SDLC erreichen.

Sie benötigen eine umfassendere, dynamischere und flexiblere Methode, um sicherzustellen, dass die Qualität während der gesamten Softwareentwicklung und -bereitstellung berücksichtigt wird und dass Fehler von Anfang an erkannt werden, statt die Verifizierung am Ende des Lebenszyklus anzusiedeln. Deshalb müssen wir uns auf den Weg zum kontinuierlichen Testing machen.

WAS IST DER WEG ZUM KONTINUIERLICHEN TESTING?

Kontinuierliches Testing bedeutet, das Testing in jede Aktivität im SDLC einzubeziehen, um unerwartetes Verhalten zu erkennen und zu beheben, sobald es entsteht.

Kontinuierliches Testing bedeutet, das Testing als grundlegenden und fortlaufenden Aspekt jeder Aktivität in den gesamten Anwendungslebenszyklus einzubetten, von den Anforderungen bis zur Produktion, um sicherzustellen, dass der erwartete geschäftliche Nutzen erreicht wird.

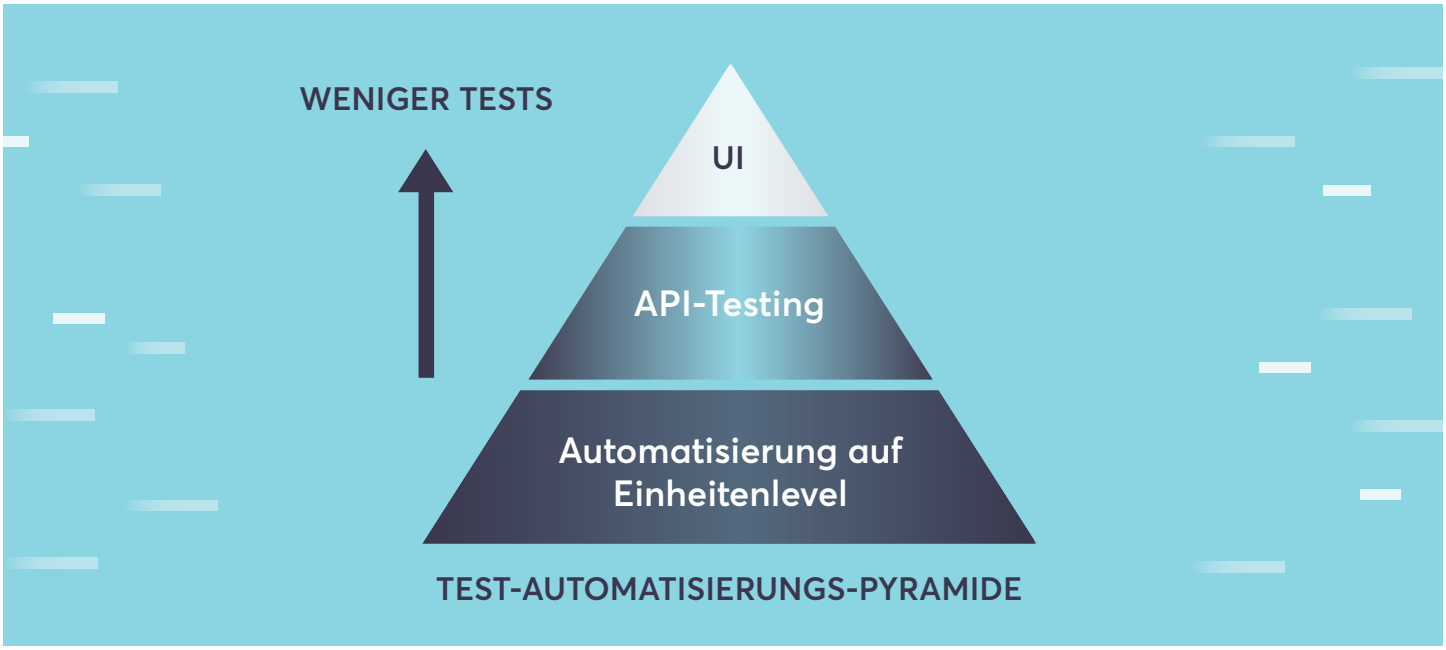
KAPITEL 2 HERAUSFORDERUNGEN FÜR DIE ERREICHUNG HOHER QUALITÄT

Die IT-Abteilung hat es noch nie leicht gehabt. Sie ist voller Fachleute und begeisterter Techniker, die sich bemühen, ein Produkt zu entwickeln und zu erstellen, das der Spezifikation entspricht – einer Spezifikation, die nicht unbedingt immer zu den Erwartungen und Grenzen ihrer menschlichen End User passt. Die Aufgaben der Software haben sich verändert. Die Produkte müssen auf vielen unterschiedlichen Plattformen zur Verfügung stehen und immer wieder andere Services bereitstellen.

Die Qualität leidet, wenn die Personen getrennt arbeiten, die die Software testen, entwickeln und die Anforderungen an sie definieren. Die IT sollte nicht mehr als eine Back-Office-Abteilung angesehen werden. Sie muss in die Entscheidungsfindung einbezogen werden.

Die Qualität leidet auch, wenn die Zeitplanung häufig unrealistisch ist und wenn Funktionen in ein Release hineingezwungen werden, ohne die Qualität mithilfe geeigneter Messdaten sicherzustellen.

Früher arbeitete das Testing als Silo. Das Ergebnis war ein deutlicher Mangel an Robustheit. Tester und das Unternehmen fragen: „Wie gut ist mein Testing? Wie gut ist meine funktionale Testabdeckung? Was ist meine funktionale Testabdeckung im Gegensatz zur reinen Codeabdeckung? Was sind meine Akzeptanzkriterien? Teste ich wirklich effektiv im Hinblick auf diese Kriterien?" Viele Tester kennen die Antworten nicht.



Um schnell für Qualität zu sorgen, müssen Tester sich bemühen, viele Tests auf Einheitenlevel, weniger Tests auf API-Level und noch weniger Tests auf UI-Level durchzuführen. Im Endeffekt sollte das Testing parallel oder gleichzeitig vorgesehen werden (Funktionalität, Performance und Security sollten zugleich getestet werden). Dann sollte der „Shift Left" erfolgen, damit es von Beginn des Lebenszyklus an kontinuierlich durchgeführt wird.

Allzu häufig verbringen Mitarbeiter Zeit mit wertlosen Aktivitäten, indem sie falsche Aspekte testen oder falsche Testdaten verwenden. Auch wenn ein Team von acht Personen mit sich zufrieden ist, müssen im Allgemeinen 40 % seiner Arbeit erneut getestet und Fehler darin behoben werden. Dies bedeutet, dass ein Team von acht Personen die Arbeit von vier Personen leistet, und häufig ist es damit einige Monate im Rückstand. Das Team muss eine Wertstromanalyse durchführen, um die richtige Herangehensweise an das Testing sicherzustellen. Die Wertstromanalyse ist eine Methode, um aktuelle Prozesse zu analysieren und einen besseren, effektiveren Prozess zu identifizieren, der der Lean-Management-Methode entspricht.

Die meisten Unternehmen verwenden „Legacy-Tools". Diese reichen aus, um nach dem Wasserfallmodell zu arbeiten, aber ein „Shift Left" des Testing unterstützen sie nicht ausreichend. Sie verhindern ein Testing in Agile-Geschwindigkeit. Sie unterstützen keine Automatisierung, die für eine Umgebung mit kontinuierlicher Integration geeignet wäre. Außerdem weigern sich Entwickler, mit diesen Tools zu arbeiten. Neue Tools sind notwendig, die die Geschwindigkeit und die Qualität nicht behindern; Tools, die es ermöglichen, die Quality Assurance in den Prozess zu integrieren und die Anwendung nicht wie bisher „nur" zu testen.

Nicht nur die Testing-Automatisierung muss jedoch verändert werden, sondern auch der Definitionsprozess für Anforderungen benötigt Verbesserungen. Anforderungen müssen eindeutig, vollständig und testbar sein.

Zurzeit entstehen 64 % der Fehler in der Anforderungsphase.

Sehr häufig werden Anforderungen auf einem abstrakten Level spezifiziert, der für Entwickler und Tester ausreicht, die sich mit dem Thema auskennen, sodass sie die Anforderungen interpretieren und die Lücken darin füllen können. Dies führt jedoch dazu, dass Teammitglieder dieselben Anforderungen unterschiedlich interpretieren. So entstehen Fehler zu Anfang des Lebenszyklus, bevor eine einzige Codezeile geschrieben wird. Im anderen Extrem können Anforderungen extrem detailliert spezifiziert werden und viele Dutzende Blatt Papier füllen. In diesem Fall dauert es zu lange, die Anforderungen zu spezifizieren, und sie können später sehr schwer gepflegt werden. Außerdem fällt es Entwicklern und Testern schwer, so viele Informationen zu überblicken, die in einer riesigen Anforderung zusammengefasst sind. Eine bessere Methode für die Spezifizierung von Anforderungen muss verwendet werden – eine, die für Agile-Teams funktioniert, die DevOps einführen.

Auch die Quality Assurance muss anders angegangen werden. Sie muss bei den leitenden Führungskräften beginnen. Diese müssen den Fokus auf der Qualität und nicht nur auf der Bereitstellung halten, da es jetzt um die gesamte User/Customer Experience geht, nicht nur um die Softwarebereitstellung oder -releases. Dies schafft die Voraussetzungen dafür, die Bereitstellung an den richtigen Anforderungen auszurichten, mit denen Entwickler und Tester verstehen können, was sie erstellen. Da Releases heute monatlich stattfinden, sind geeignete und effiziente Feedbackschleifen unverzichtbar. Die QA darf nicht mehr als Element zweiter Klasse angesehen werden oder sich so verhalten, nur weil sie bisher als gemeinsame Serviceverantwortlichkeit identifiziert wurde.

Um diese Veränderung zu erreichen, müssen die Unternehmenskultur und die Organisation verändert werden. Hierbei geht es um Personen, nicht um Technik; es ist jedoch für einen erfolgreichen Weg zum kontinuierlichen Testing unverzichtbar. Mit jeder technologischen Veränderung, wie dem Übergang zum kontinuierlichen Testing oder dem „Shift Left", können manche Personen gut umgehen und andere nicht. Die Veränderung bedroht auch die Centers of Excellence (CoE), da Tester infrage stellen, ob das „Testing" ungeschulten Personen überlassen werden kann.

Dies ist der Hauptgrund dafür, dass ein CoE von einem „Center of Excellence" in ein "Center of Enablement" (Schulungs- und Unterstützungszentrum) transformiert werden muss. Die Personen, die jetzt die Verantwortung für das Testing im gesamten SDLC übernehmen müssen, müssen die geeigneten Prozesse kennen und über die richtigen Tools verfügen, um sie leichter durchzuführen. Die Tester mit der größten Kompetenz und Erfahrung müssen sie dabei unterstützen. Das Change Management muss hierbei top-down ebenso wie bottom-up unterstützt werden.

Der „Shift Left" des Testing erfordert auch einen „Shift Left" der Performance- und Security-Tests. Diese sollten nicht mehr anders als die Funktionstests behandelt werden. Stattdessen müssen sie mit dem Funktions-Testing auf eine

Stufe gestellt werden. Zurzeit reicht das Security-Testing vor der Bereitstellung in der Produktion nicht aus. Es ist an der Zeit, die Security als Element erster Klasse zu behandeln und spezielle Security-Testing-Praktiken in den Prozess des kontinuierlichen Testing aufzunehmen.

Nachdem eine Anwendung in die Produktion gegangen ist, kann sich herausstellen, dass sie mit tatsächlichem Anwenderdatenverkehr nicht die vorgesehene Performance erbringt. Eine passende schnelle Abhilfe scheint darin zu liegen, mehr Ressourcen zuzuordnen. Dies ist jedoch nur in einer Cloud-basierten Infrastruktur möglich. Daher muss sichergestellt werden, dass die Application Performance getestet wird, bevor die Anwendung in die Produktion geht. Dies muss von Anfang an geschehen: auf Einheitenlevel durch die Entwickler und danach schrittweise in größerem Umfang, wenn Codeeinheiten zu immer größeren Komponenten zusammengefügt werden, die Storys, Funktionen und Epics umsetzen. Dabei müssen stets die zuvor erstellten kleineren Performance-Tests genutzt werden. Unternehmen müssen geeignete Tools und Prozesse verwenden, damit dies möglich ist.

Eine Veränderung der Unternehmenskultur erfordert Top-down- und Bottom-up-Veränderungsinitiativen in Verbindung mit hervorragenden Führungsqualitäten. Insgesamt ist eine Unternehmenskultur notwendig, die zum kontinuierlichen Testing passt. Führungskräfte und IT-Manager müssen mit Testern und Entwicklern sprechen und sich darauf konzentrieren, gemeinsam einen Plan zu erstellen. Machen Sie diese Personen zu einem Teil der Lösung, indem Sie sie in die Initiative einbeziehen. Geben Sie ihnen die Chance, Fehler zu machen und daraus zu lernen.

KAPITEL 3 WAS KONTINUIERLICHES TESTING AUSMACHT

Kontinuierliches Testing bedeutet, dass alle Personen programmieren und testen. Die Teammitglieder sind jeder für sich und alle gemeinsam für die Entwicklung und Bereitstellung hochwertiger Software verantwortlich. Dies zeigt eine Weiterentwicklung vom Wasserfallmodell über den Scrum-Ablauf, die Agile-Methode sowie die Verschmelzung von Agile und DevOps bis zur Business Agility, unterstützt durch kontinuierliches Testing.

Zur Ausübung der gemeinsamen Verantwortung werden Rückblicke verwendet, um den Prozess und seine Ergebnisse zu untersuchen und anzupassen. Damit wird die Definition von „fertig" neu festgelegt. Um eine Denkweise zu erreichen, die das kontinuierliche Testing unterstützt, müssen Entwickler schnell Feedback erhalten, damit sie sicherstellen können, dass ihr Code funktioniert. So kann der Code die Releasepipeline schnell durchlaufen. Hierfür müssen Testdaten für alle Funktionstests und sonstigen Tests erstellt oder gefunden werden, die vor und in der Produktion durchgeführt werden.

Die drei wichtigsten Elemente für das kontinuierliche Testing sind:

PERSONEN

Stellen Sie Entwicklern Testing-Funktionen zur Verfügung. Entwickler und Agile-Teams müssen sich bewusst sein, dass das Testing ein Teil ihrer Verantwortung ist.

PROZESSE

Automatisieren Sie so viel wie möglich.

TECHNOLOGIEN

Stellen Sie Testing-Lösungen bereit, die die Entwickler wirklich verwenden. Dies bedeutet Open Source in Form von Code in der Cloud.

Ein Unternehmen muss sicherstellen, dass es über eine Definition des kontinuierlichen Testing verfügt, die von allen auf die gleiche Weise verstanden wird. Es ist noch relativ neu, und wie in vielen anderen Bereichen der Softwareentwicklung kommen unterschiedliche Definitionen und Praktiken vor.

Kontinuierliches Testing bedeutet, das Testing in jede Aktivität im SDLC einzubeziehen, um unerwartetes Verhalten zu erkennen und zu beheben, sobald es entsteht.

Ein guter Vergleich ist die Vorstellung von einem automatisierten Softwarefließband, das in jedem Schritt ein Testing umfasst, angefangen bei der Erfassung der Anforderungen und jeweils automatisch ausgelöst vom ersten Einchecken von Code bis in die Produktion. Dazu gehören alle Aspekte des Testing bis hin zum Integrations- und Performance Testing und Monitoring in der Produktion. Das kontinuierliche Testing sollte ein effektives und effizientes, anwendungszentriertes End-to-End-Testsystem darstellen – für die kontinuierliche Integration (CI), die kontinuierliche Entwicklung (CD) und die Produktion. Sein Ziel besteht darin, Änderungen so effizient wie nur möglich in die Produktion zu bringen, mit kürzester Vorlaufzeit, höchster Qualität und einer überragenden Customer Experience.

KAPITEL 4 DIE DISZIPLINEN DES KONTINUIERLICHEN TESTING

Um einen Code-Release-Prozess durch kontinuierliches Testing zu beschleunigen und zu verbessern, muss mit einer Reihe von Disziplinen sichergestellt werden, dass die Qualität Schritt hält. Um die Qualität von Anwendungen sicherzustellen, müssen Teams modernere Testing-Praktiken in Form kleiner Qualitätsprüfungen hinzufügen, die während der gesamten Anwendungspipeline durchgeführt werden. Dies ermöglicht ein fortlaufendes Testing kleiner Codeabschnitte.

Die Testautomatisierung ist nicht immer die erste Aufgabe, die in Angriff genommen werden sollte. Der Grund hierfür ist, dass auch nach der Automatisierung von API- und GUI-Tests und ihrer Integration in einen Agenten für die kontinuierliche Integration noch Abhängigkeiten für diese Tests vorliegen – Testdaten, Schnittstellen und Umgebungen – die vor der Testdurchführung berücksichtigt werden müssen. Gehen Sie wie folgt vor:

- Beheben Sie umgebungsbedingte Einschränkungen, und geben Sie Entwicklern und Testern genug Zeit, ihre Aufgaben zu erledigen, sogar, wenn sie dies manuell tun. Virtualisieren Sie alle Schnittstellen, die Sie nicht besitzen, über die Sie nicht die Kontrolle haben oder die Sie nicht testen möchten.
- Verwenden Sie temporäre Testumgebungen.
- Automatisieren Sie die Provisionierung und das Management von Testdaten als On-Demand-Grundlage für Ihre manuellen oder automatisierten Tests.
- Automatisieren Sie Ihre Tests, damit sie mithilfe geeigneter Testdaten und Ihrer virtualisierten Schnittstellen ausgeführt werden können.
- Richten Sie die Orchestrierungs-Engine für Ihre Pipeline so ein, dass keine manuellen Eingriffe in die obigen Schritte erforderlich sind.
- Konzentrieren Sie sich dann auf die ergänzenden Disziplinen.

ELF DISZIPLINEN DES KONTINUIERLICHEN TESTING

1. **VIRTUALISIERTE UMGEBUNGEN**
   Kontinuierliches Testing bedeutet häufigeres Testing. Dies bedeutet, dass Sie mehrere Umgebungen häufiger verwenden werden. Und das stellt ein Problem – einen Engpass – dar, wenn diese Umgebungen nicht stets verfügbar sind. Einige Umgebungen sind über APIs zugänglich, andere über verschiedene Messaging-Schnittstellen. Diese Umgebungen wurden möglicherweise mithilfe moderner Architekturen entworfen, während es sich bei anderen um monolithische Legacy-Client/Server- oder -Mainframe-Systeme handelt. Wie koordinieren Sie also das Testing mit mehreren Umgebungsverantwortlichen, die ihre Umgebungen nicht notwendigerweise immer am Laufen halten, damit Sie sie für Tests verwenden können? Indem Sie diese Umgebungen virtualisieren, können Sie Ihren Code testen, ohne sich um Bereiche kümmern zu müssen, die Sie nicht verändern (d. h. andere Systeme und Umgebungen). Indem Sie diese Umgebungen als temporäre Elemente on demand per Virtualisierung bereitstellen, beheben Sie die entsprechende Einschränkung für Ihren Entwicklungszyklus, da sie so immer verfügbar sind. Sie haben die Kontrolle über sie und können sie bei Bedarf einrichten.

2. **TESTDATENMANAGEMENT**
   Sie benötigen die richtigen Daten und die richtige Vielfalt an Daten, um positive und negative Szenarien testen zu können. Diese Vielfalt finden Sie nicht in der Produktion, und das ist ein kritisches Problem. Sie müssen daher synthetische Daten erzeugen, um ein kontinuierliches Testing zu erreichen und dabei sicher zu sein, dass keine personenbezogenen Daten in den Testdaten vorhanden und damit gefährdet sind. Außerdem können Sie Produktionsdaten nicht so schnell erfassen, bearbeiten und provisionieren, wie Ihre Anwendungspipeline es erfordert.

3. **TESTAUTOMATISIERUNG**
   In einer vollständig orchestrierten Anwendungspipeline werden bestehende Scripts aus Gründen, die nichts mit dem Anwendungscode zu tun haben, fehlerhaft ausgeführt. Daher traut niemand den Ergebnissen. Sie benötigen eine zuverlässige Testautomatisierung, um kontinuierliches Testing zu erreichen.

4. **PIPELINEORCHESTRIERUNG**
   Das Rückgrat. Mit ihm ist alles verbunden. Es muss in Ihre Automatisierungssuite integriert sein. Sie müssen verstehen, wie es funktioniert, wie Sie Ergebnisse interpretieren können und wie Sie es skalierbar machen. Damit richten Sie das kontinuierliche Testing ein, indem sie entsprechende Aspekte in eine Pipeline integrieren. Stellen Sie sicher, dass dies transparent ist und dass alle Personen vollständig sehen können, was die Pipeline durchläuft. Dieses automatisierte Workflow-Tool führt alle Ihre automatisierten Tests aus und ist beim Durchlaufen der Pipeline vollständig in Codebereitstellungsaktivitäten integriert. Im Rahmen einer DevOps-Einführungsinitiative wäre es unrealistisch, zu erwarten, dass kontinuierliches Testing ohne die Zuverlässigkeit und Geschwindigkeit einer standardisierten und automatisierten Pipeline erreicht werden können.

5. **API-TESTING**
   Dies ermöglicht die Koordination der Testing-Pyramide. Unternehmen sollten sich bemühen, so viel wie möglich auf Einheiten- und API-Level zu testen und sich möglichst wenig auf das UI-Testing zu stützen. Um ein kontinuierliches Testing zu erreichen, müssen Sie das Konzept der Testing-Pyramide geeignet umsetzen, das Einheiten- und API-Testing stärken und die Notwendigkeit des UI-Testing reduzieren, vor allem zum Testen von Geschäftslogik.

6. **PERFORMANCE-/LAST-TESTING**
   Auch wenn die Funktionalität von Anwendungen in Ordnung ist, müssen Sie Ihren Denkansatz für das Performance Testing radikal verändern. Ein „Shift Left" des Testing bedeutet, dass Sie früher im Lebenszyklus testen und weniger Anwendungs- und Infrastrukturkomponenten einbeziehen, sodass die Tests von sich aus kleiner sind. Es sind jedoch auch viel mehr. Machen Sie diese Möglichkeit allen Personen in allen Agile-Teams zugänglich. Dies bedeutet, dass alle Entwickler und Tester gleichermaßen in der Lage sein müssen, einen Performance-Test zu erstellen und selbst durchzuführen, ohne Anfragen an andere Personen zu senden.

7. **SECURITY-TESTING**
   Entwickler müssen genau wissen, in welchem Zustand der Code zur Veröffentlichung zugelassen wird und in welchem nicht. Sie benötigen geeignete Schulungen und Tools, um zu messen, wie sicher ihr Code ist. Entwickler benötigen fortlaufende Schulungen, da sich der Bereich der Anwendungs-Security stets weiterentwickelt. Die Anwendungspipeline muss so eingerichtet sein, dass sie automatisch statische Analysen, dynamische Analysen und Analysen der Softwarezusammensetzung durchführt, sodass alle Security-Probleme erkannt und hervorgehoben werden. Diese müssen dann als Chance genutzt werden, die Entwickler weiter zu schulen. Ein Security-Testing muss in jedem Schritt des Lebenszyklus durchgeführt werden.

8. **EINFÜHRUNG TESTGETRIEBENER UND VERHALTENSGETRIEBENER ENTWICKLUNG**
   Betrachten Sie die Akzeptanzkriterien für jede User Story, und erstellen Sie die notwendigen Tests, um sie zu erfüllen. Dies wahrt den Fokus des Testing in den Sprints und sorgt dafür, dass die Entwickler sich an den Unternehmenserwartungen orientieren. Im Laufe der Zeit definieren Teams wesentlich detailliertere Akzeptanzkriterien, sodass auch die Tests umfassender sein müssen.

9. **AUTOMATISIERTE TESTERZEUGUNG**
   Manuelle Tests oder automatisierte Test-Scripts manuell zu entwerfen und zu schreiben, stellt einen Engpass dar. Erzeugen Sie die neuen Akzeptanzleveltests für das manuelle und das automatisierte Testing automatisch zu Anfang des Sprints. Sobald Entwickler beginnen, ihre ersten funktionierenden Codeeinheiten einzuchecken und zusammenzuführen, können die Tests somit automatisiert ausgeführt werden. Die Entwickler können sofortiges Feedback zur Qualität ihres Codes erhalten, sobald sie ihn in die Versionsverwaltung einchecken.

10. **UMGANG MIT ANFORDERUNGEN**
    Sie müssen alle SDLC-Stakeholder in den Weg zum kontinuierlichen Testing einschließen. Dies bedeutet, dass Sie eine bessere Möglichkeit finden müssen, über die Anforderungen zu kommunizieren und zusammen an ihnen zu arbeiten. Je früher Sie den Umgang mit Anforderungen in Ihre Initiative für das kontinuierliche Testing einbeziehen, umso reibungsloser ist Ihr Weg zum kontinuierlichen Testing, da alle Teammitglieder die Anforderungen von Anfang an auf dieselbe Weise verstehen.

11. **FEEDBACKSCHLEIFEN**
    Diese sind für ein erfolgreiches kontinuierliches Testing unverzichtbar. Dashboards müssen automatisch in Echtzeit verfügbar sein, und das gesamte Team muss Zugang zu ihnen haben. Feedbackschleifen im gesamten SDLC, nicht nur in der Produktion, sollten als Kompass verwendet werden, mit dem Sie durch Ihre Transformation zum kontinuierlichen Testing navigieren.

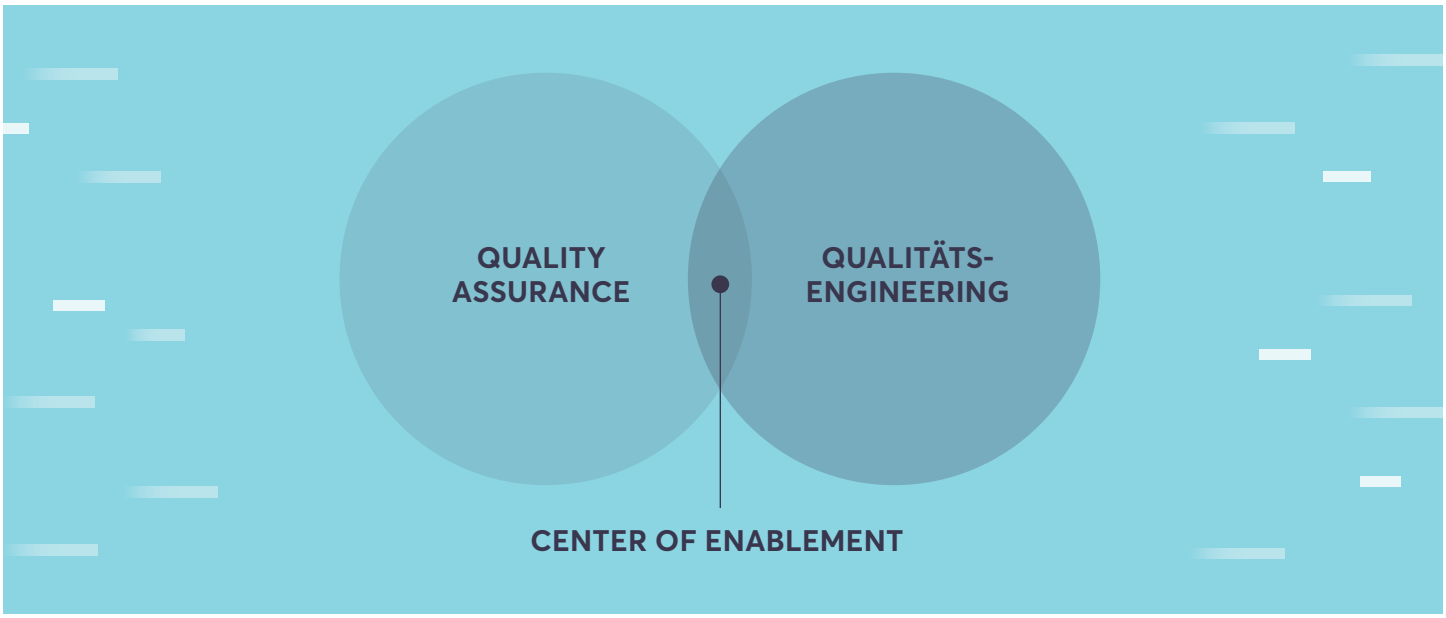
KAPITEL 5 BESCHLEUNIGEN SIE IN DEN KURVEN

Der Übergang zu einer Unternehmenskultur des kontinuierlichen Testing ist eine wichtige Komponente der Einführung von DevOps, aber er bedeutet auch, sich in einen neuen Wissens- und Praxisbereich einarbeiten zu müssen. Dies ist ein wenig gewöhnungsbedürftig, aber die Vorteile sind durch Theorie und Praxis belegt. Dass derartige Verbesserungsinitiativen dringend notwendig sind, zeigt auch die belegbare Tatsache, dass Kunden im Einzelhandels- und B2B-Bereich sehr wenig Geduld mit einer schlechten Customer Experience haben. Dies hat weit schnellere und leichter sichtbare Einflüsse auf den Unternehmenserfolg.

Natürlich ist es gut möglich, dass einige dieser neuen Praktiken der vorhandenen Unternehmenskultur zuwiderlaufen, aber das haben Veränderungen und Innovationen nun mal so an sich. Teams müssen an diese neuen Praktiken herangeführt und geschult oder unterstützt werden, damit sie die Transformation erfolgreich durchlaufen können.

Beispielsweise sollten bei der Programmierung einer Funktion Release-Hinweise in den Code eingebettet werden, damit sie für alle lesbar sind. Erstellen Sie – genau wie im Projektmanagement üblich – einen Plan, der so detailliert ausgearbeitet ist, dass jemand anders bei Bedarf übernehmen kann.

Die Quality Assurance (QA) sollte sich zum Qualitätsengineering als Center of Enablement weiterentwickeln. Dort integrieren Teams das Wissen dazu in die Software, wie sie getestet, bereitgestellt, überwacht und sogar automatisch korrigiert werden kann, sodass Anwendungsteams diese Informationen in ihren Scrums nutzen können, ohne von externen Mitarbeitern abhängig zu sein.



Alles sollte als Code behandelt werden, sodass es auf standardisierte und skalierbare Weise durch CI unter Kontrolle gehalten werden kann. Hierfür müssen Teams darüber nachdenken, wie der Code getestet wird, um sicherzustellen, dass er bei Integration in das Quellcode-Repository erwartungsgemäß arbeitet und keine negativen Auswirkungen auf andere Teile der Anwendung hat.

Die beteiligten Personen, also Entwickler und Tester, müssen Teil dieser neuen Unternehmenskultur werden. Eine DevOps-Abteilung sollte eingerichtet werden, um Personen einzustellen oder zu schulen, die als vielseitigere Techniker mehrere Rollen ausfüllen können. Hierzu kann u. a. eine Matrixorganisation verwendet werden, mit der das Anwendungsentwicklungsteam die notwendigen Fähigkeiten bei Bedarf einwerben oder nutzen kann. Das Change Management für Personen ist genauso erfolgskritisch wie die physische Transformation zum kontinuierlichen Testing.

Die Cloud als Plattform wird zu einem enormen Erfolgsfaktor. Ein Weg zum kontinuierlichen Testing kann nicht allein mit On-Premise-Systemen und Legacy-Systemen gegangen werden.

KAPITEL 6 WO STEHE ICH AUF DEM WEG ZUM KONTINUIERLICHEN TESTING?

Schon früh auf dem Weg zum kontinuierlichen Testing kommt ein Punkt, an dem der Leiter und die Stakeholder innehalten und sich fragen müssen: „Wo stehe ich in der Zeitplanung für die Bereitstellung?". Es ist unerlässlich, zuerst zu verstehen, warum kontinuierliches Testing notwendig ist, und es dann passend zu definieren. Bei der Vorbereitung auf die Einführung muss jedoch auch die Frage gestellt werden, wo das Team zurzeit steht.

ZWEI WICHTIGE ASPEKTE, DIE AN DIESEM PUNKT BEACHTET WERDEN SOLLTEN

- Stellen Sie den Kunden stets ins Zentrum des Veränderungsprozesses und der daraus resultierenden kontinuierlichen Testing-Aktivitäten.
- Ermitteln Sie, wie Sie die Qualität geeignet messen können.

An diesem Punkt müssen neue Techniken eingeführt werden, und Erfolgshindernisse müssen identifiziert und überwunden werden.

EINZUFÜHRENDE TECHNIKEN

1. **VERBESSERN SIE DIE BEZIEHUNG ZWISCHEN ALLEN TESTERN UND ENTWICKLERN**
   Halten Sie die Teams klein, fördern Sie die Zusammenarbeit zwischen Teams, und erleichtern Sie den Zugang zu Berichten und ihre gemeinsame Nutzung online.

2. **WEISEN SIE DER AUTOMATISIERUNG HOHE PRIORITÄT ZU**
   Streichen Sie nicht alle manuellen Testing-Aufgaben, aber führen Sie eine Denkweise ein, in der die Automatisierung an erster Stelle steht. Konzentrieren Sie sich auf Bereiche, die wiederholt ausgeführt werden. Richten Sie also zuerst die Pipeline ein, damit Anwendungen sie später ohne Sorgen oder Angst vor Lecks durchlaufen können. Zeichnen Sie Ihren SDLC auf, und identifizieren Sie darin Möglichkeiten für die Automatisierung.

3. **ARBEITEN SIE IN KLEINEN SCHRITTEN**
   Teilen Sie Arbeiten in kleinere Inkremente auf, die nach dem Entwurf und der Programmierung leichter getestet werden können. Dies erleichtert die Automatisierung der Tests und die Bereitstellung.

4. **VERFOLGEN SIE ALLES**
   Verwenden Sie Messdaten und festgelegte Erfolgskriterien. Beim kontinuierlichen Testing geht es darum, sofort zu identifizieren, ob etwas funktioniert oder nicht. Stellen Sie daher sicher, dass dies leicht möglich ist.

5. **BESCHAFFEN SIE SICH DIE RICHTIGEN TOOLS FÜR DAS KONTINUIERLICHE TESTING**
   Finden Sie die Tools, die Ihnen helfen, kontinuierlich zu entwickeln, zu testen und zu analysieren. Wählen Sie führende Tools aus, die zusammenarbeiten und leicht in Ihre Arbeitsumgebung integriert werden können. Entscheiden Sie sich für Tools, die von der Community unterstützt werden, indem alle Personen ihre Herausforderungen, Lösungen und interessanten Anwendungsfälle mitteilen.

6. **ENTWICKELN SIE EIN SYSTEM, UM IHRE ERGEBNISSE ANZUZEIGEN**
   Sehen Sie sich Ergebnisse detailliert an, denn so erfahren Sie, ob Ihr Code funktioniert und wo die Lücken sind. Definieren Sie Ihre KPIs und Akzeptanzkriterien, und machen Sie sie quantifizierbar. Erstellen Sie Dashboards, um die KPIs zu verfolgen. Zeigen Sie dabei ein Grundverhalten und nachfolgende Änderungen an.

Der Übergang zum kontinuierlichen Testing kann mühsam erscheinen. Am besten erreichen Sie ihn jedoch, indem Sie mit einem kleinen Projekt anfangen, es erfolgreich abschließen und dann mit anderen kleinen Projekten fortfahren. So entsteht ein iterativer Prozess, mit dem Sie zunächst kleine Erfolge anstreben und dann auf diesen aufbauen, um Schwung und Wissen zu vergrößern.

Es wird auch Rückschläge geben, aber die Teams und die Leitung sollten darauf vorbereitet sein, Misserfolge zu begrenzen und sie zu nutzen, um aus der Erfahrung zu lernen.

Es ist sehr wichtig, dass Führungskräfte Verantwortlichkeiten im Unternehmen zuweisen. Ein einzelner Vordenker kann nur 20–30 % der Beteiligten erreichen. Für den Rest des Unternehmens müssen die Führungskräfte Erwartungen definieren und den Kurs halten.

Beim Übergang zum kontinuierlichen Testing treten auch Hindernisse auf, und diese müssen einzeln identifiziert werden. Dazu kann die Unmöglichkeit gehören, auf Open-Source-Lösungen zuzugreifen und sie zu nutzen. Die Unternehmenskultur – nicht nur Personen, sondern auch Praktiken – ist möglicherweise nicht ausreichend vorbereitet. Es ist wahrscheinlich, dass die Zeit nicht ausreicht, um mit der Umsetzung, den Schulungen und dem Übergang zum kontinuierlichen Testing zu beginnen. Hinzu kommen möglicherweise ineffiziente Führungsmethoden und das Vorhandensein von Legacy-Anwendungen.

Das wesentliche Ziel dieser Phase besteht darin, ein Reifegradmodell für das kontinuierliche Testing zu entwickeln und einzuführen. Hierfür müssen die richtige Denkweise, die richtigen Herangehensweisen und die richtigen Fähigkeiten aufgebaut werden. Richten Sie die Testautomatisierung für das Funktions- und Security-Testing sowie Performance Testing auf Einheiten- und API-Level ein, und halten Sie die Testautomatisierung auf UI-Level möglichst klein, da Scripts sehr empfindlich auf Veränderungen der UI reagieren. Fügen Sie eine Feedbackschleife hinzu, die Performance- und Anwender-Monitoring-Tools in Vorproduktionsumgebungen als Teil der Pipeline einbezieht. Eine vollständige Telemetrie beschleunigt die Behebung von Fehlern.

Alle diese Funktionen müssen vollständig integriert und zusammenarbeitsorientiert sein und dürfen sich in keinerlei Silos befinden. Unternehmen müssen in der Lage sein, Open Source zu akzeptieren und zu verstehen, da dies Personen gestattet, zu experimentieren und Fehler früher zu erkennen, ohne Ausgaben rechtfertigen zu müssen. Außerdem sollten sie für geeigneten Zugang zu Testdaten sorgen, da ihr Fehlen eines der größten Hindernisse für den erfolgreichen Abschluss ist.

KAPITEL 7 MESSDATEN

Jedes Projekt erfordert Messdaten, um Fortschritte und Qualität zu ermitteln und den geschäftlichen Nutzen zu identifizieren. Das kontinuierliche Testing bildet hier keine Ausnahme. Das kontinuierliche Testing könnte sogar als „kontinuierliche Validierung" bezeichnet werden, da Sie stetig überprüfen, ob Ihre Anwendung oder Ihr Service den erwarteten geschäftlichen Nutzen bereitstellen wird (oder bereitstellt). Einige der übergreifenden Benchmarks sind die Kundenakzeptanz, die Kundeninteraktion und der Umsatz. Teams sollten sich bemühen, die Meinungen von End Usern zu verstehen und zu quantifizieren und sie in eine kontinuierliche Feedbackschleife für die Entwickler und das Unternehmen aufzunehmen.

Außerdem müssen Sie in der Lage sein, zu verfolgen, wie viele Probleme vorliegen – d. h. wie viele Fehler in der Produktion und Vorproduktion – und dies mit der Geschwindigkeit in Einklang bringen, in der Sie bereitstellen können. Wenn Personen zunehmend besseren Code schreiben, sollte sich das Verhältnis verändern: Es sollte weniger Fehler bei konstanter Release-Kadenz geben oder noch besser bei schnellerer Release-Kadenz.

Kombinieren Sie die Codeabdeckung und die erforderliche Abdeckung (Funktionalitätsabdeckung), um die gesamte Funktionalitätsabdeckung zu verfolgen.

Stellen Sie sicher, dass Sie verstehen, was Qualität für Ihr Unternehmen bedeutet, und legen Sie ein Grundverhalten fest. Wenn Sie nicht wissen, was Ihr Ausgangspunkt ist, können Sie nicht erkennen, ob Sie sich verbessern. Verwenden Sie einen KPI, der die Produktionsqualität misst.

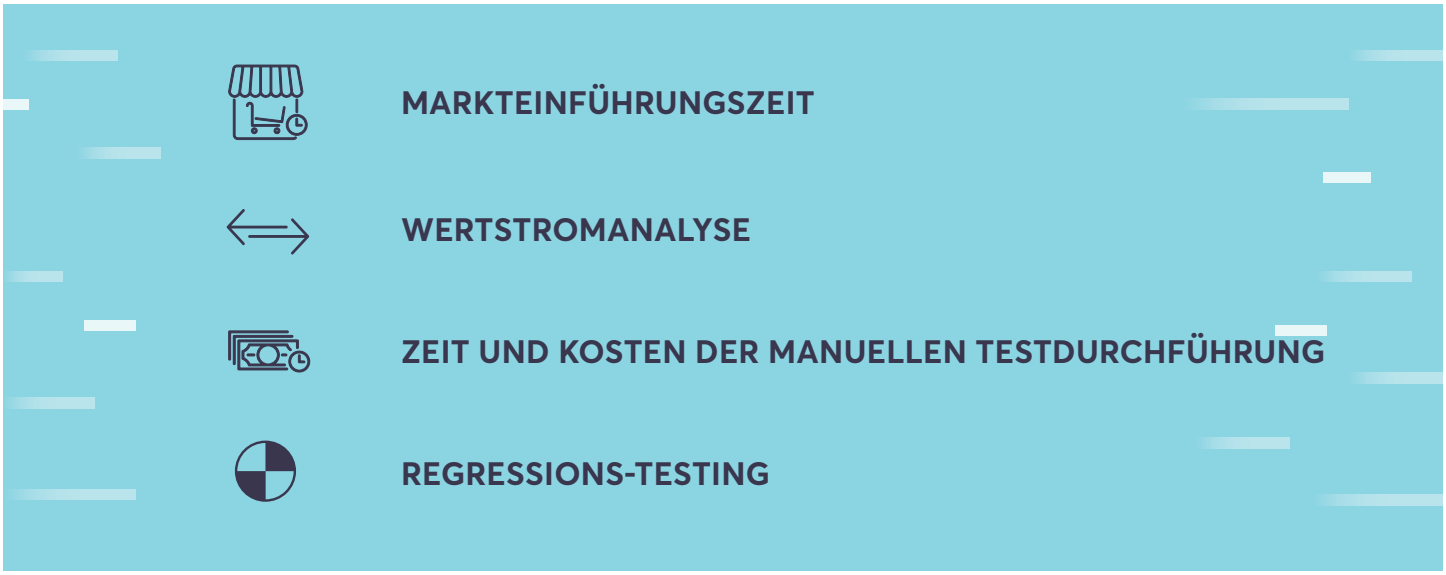
Erfahren Sie, welche Art von Abdeckung für Einheitentests, Funktionstests, Performance und Security erforderlich ist. Sie sollten nicht mehr zwischen Funktions-Testing und sonstigem Testing unterscheiden.

Messen Sie die Gesamtzeit bis zur Bereitstellung in der Produktion. Das Ziel des kontinuierlichen Testing besteht darin, schneller Releases von höherer Qualität bereitstellen, die geschäftlichen Nutzen bieten. Konzentrieren Sie sich auf schnellere Releases, senken Sie zugleich den Anteil der Fehler in der Produktion, und verkürzen Sie die Markteinführungszeit durch Automatisierung.

Verändern Sie die Denkweisen der Personen, damit sie nicht nur nach Erfolg oder Misserfolg unterscheiden, indem Sie eine passende Definition von „fertig" entwickeln. Fehler müssen verfolgt werden, damit die Analysen für Verbesserungen genutzt werden können.

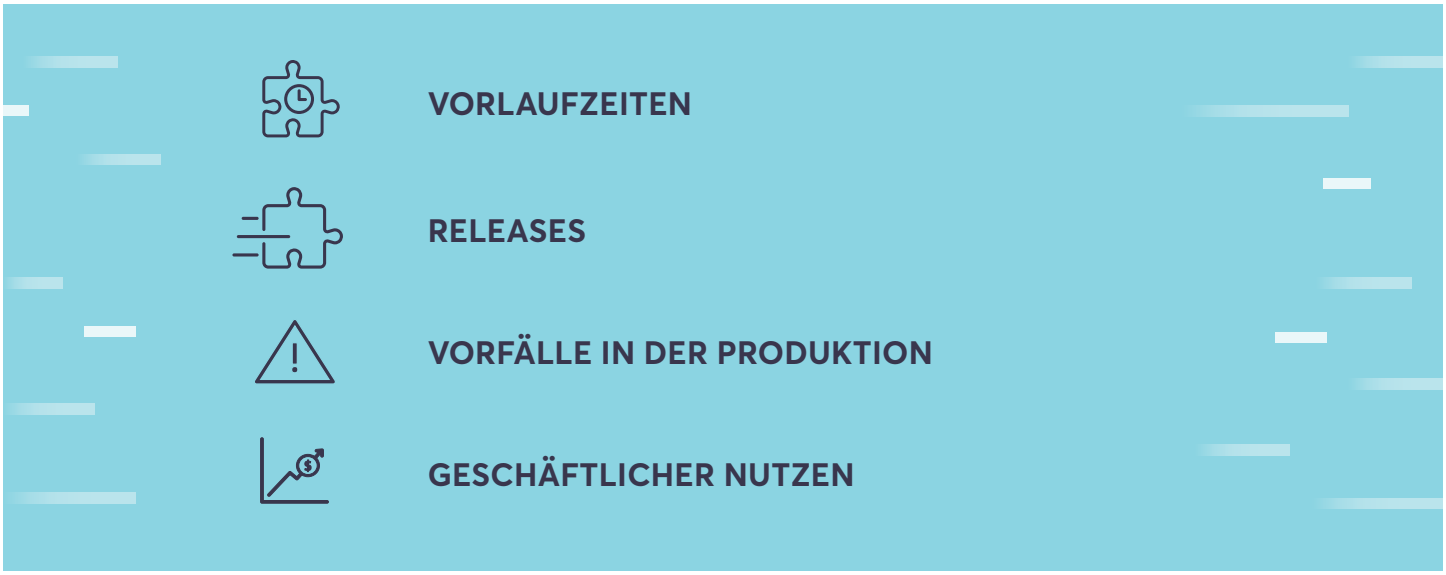
Sie können den Erfolg des kontinuierlichen Testing messen, indem Sie in den Testumgebungen auf niedrigerem Level testen. Dies kann schnell durchgeführt werden, weil die Tests kleiner sind. Achten Sie darauf, dass das Testing mit der Entwicklung Schritt halten kann und dass Sie zu jedem Zeitpunkt Code ausliefern können, nicht nur am Ende des Sprints.

Beachten Sie bei der Bewertung und Messung der Qualität und Quantität Folgendes:



Beginnen Sie den „Shift Left" zu den Entwicklern, indem Sie Testing-Tools demokratisieren und Tester zu Technikern weiterentwickeln.

Jede Führungskraft kann und soll ihre eigene Ansicht dazu haben, welche Messdaten für das kontinuierliche Testing am wertvollsten sind. Hervorragend geeignet sind folgende Messdaten:



Beim Übergang vom Wasserfallmodell zur Agile-Methode muss das Testing effizient sein, um Akzeptanz zu gewinnen. Im Idealfall entsteht diese Effizienz durch Automatisierung und „Shift Left". Beim Übergang zu immer kürzeren Release-Intervallen auf Ihrem Weg zum kontinuierlichen Testing kann es nicht nur darum gehen, die Automatisierung und den „Shift Left" zu verstärken, sondern es geht darum, ganz anders über das Testing zu denken.

Dies erfordert eine Kompetenz für das kontinuierliche Testing, die nicht zwischen CI und CD angesiedelt ist, sondern CI und CD umfasst. Sie muss durch eine Reihe von Tools unterstützt werden, die diese unterschiedlichen Verfahren und Denkweisen für das Testing unterstützen.

Sie benötigen Tools, die die Planung, die Testpläne und die Testautomatisierung unterstützen. Diese müssen in den Tool Chains für CI und CD eingesetzt werden.

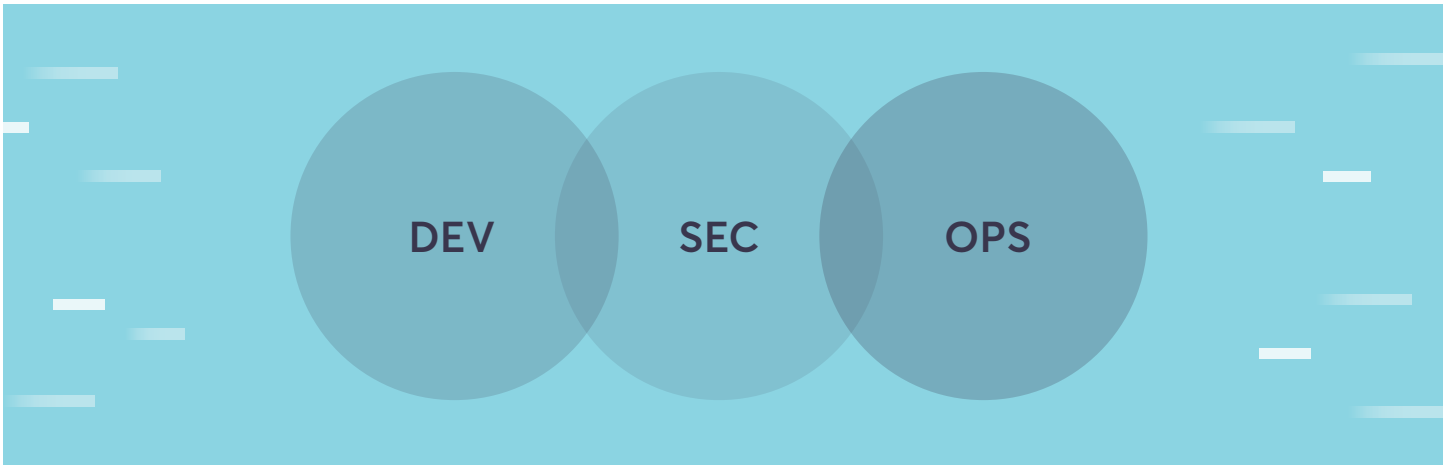
KAPITEL 8 SECURITY

Beim Entwurf der Strategie für einen Übergang zum kontinuierlichen Testing müssen Sie die Security unbedingt in den Mittelpunkt stellen. Security-Spezialisten werden zu Recht darauf hinweisen, dass die Security bisher erst ganz am Ende beachtet wurde und den Entwicklern außerdem entgegengestellt wurde. Während die Entwickler ihr Produkt aus der Tür bekommen möchten, halten die Security-Spezialisten sie stets auf.

Die Security muss von Anfang an in den Ablauf des kontinuierlichen Testing einbezogen werden. Sie hat höchste Priorität bei der Identifizierung und Bestätigung aller Aspekte der Wertkette, von den ersten Anfängen bis zur Auslieferung an den Kunden. Anhand von Feedbackschleifen kann überprüft werden, ob das Produkt die Erwartungen erfüllt, und das Team kann Security-Probleme finden und beheben, sobald sie auftreten.

Entwickler erzeugen häufig Security-Schwachstellen, ohne es zu merken. Beim kontinuierlichen Security-Testing geht es nicht nur darum, diese Schwachstellen zu beheben. Es geht auch darum, welche Schulungen den Entwicklern helfen können, die Security ihres Codes besser zu verstehen und durch höheres Bewusstsein diese Schwachstellen von Anfang an zu vermeiden. Das Security-Team muss zu einem Partner, Berater und Experten werden, dem die Entwickler vertrauen und der ihnen Schulungen und Tools mit messbaren Erfolgen bietet.

Das Security-Testing muss in jedem Schritt durchgeführt werden. Daher müssen Entwickler Security-Scans ausführen, um Probleme zu finden, sobald sie auftreten. Zurzeit sind Security und Testing von den Entwicklern getrennt. Stellen Sie sich hier keinen Konflikt zwischen Security und Qualität vor, sondern eine Partnerschaft, die auf eine gemeinsame Lösung hinarbeitet.



In den Anforderungen muss klar dargelegt werden, was ausreichende Security bedeutet. An der Security müssen alle arbeiten, aber wir müssen festlegen, wann wir damit fertig sind. Das Entwicklungsteam muss geeignete Schulungen erhalten, um diese Anforderungen zu verstehen, und geeignete Tools, um ihre Erfüllung zu messen. Das Testing mit diesen Tools sollte in unsere Pipelines aufgenommen werden, damit wir uns kontinuierlich selbst bewerten. Wie in jedem guten DevOps-Prozess dient das Security-Scanning als Feedback, das verwendet werden sollte, um den aktuellen Zustand zu messen und Schulungen passend zum „Shift Left" anzubieten, damit das Team im Laufe der Zeit seine Kompetenzen erweitert. Entwicklern zu zeigen, wie sie von Anfang an korrekt programmieren können, erhöht nicht nur die Security, sondern auch die Geschwindigkeit sehr.

Auch die Führungsebenen im Unternehmen müssen für die Security verantwortlich sein. Von dort aus gelangen die Fragen und Lösungen ganz von selbst bis auf die untersten Organisationsebenen. Wenn Sie nur auf diesen untersten Ebenen anfangen, können Sie ein paar Teams einbeziehen, aber um wirkliche Akzeptanz im gesamten Unternehmen zu erreichen, müssen Sie sich die Unterstützung der Führungskräfte sichern.

Und schließlich muss das Security-Team ebenfalls seine Denkweise weiterentwickeln, damit seine Mitglieder sich als einen integralen Bestandteil der IT ansehen.

KAPITEL 9 ZUKUNFTSFÄHIGKEIT

Ein Plan für das kontinuierliche Testing muss zukunftsfähig sein, um in einer Zeit ungebremster Veränderung und Geschwindigkeit zu bestehen. Um dies zu erreichen, benötigen Sie eine Strategie für alle Aspekte der Entwicklung. Diese muss von der Programmierung bis zum Testing alles einbeziehen, einschließlich Funktionalität, Performance und Security. Auch das Management der Umgebung gehört dazu, von der Testdatenerzeugung bis hin zur Service-Virtualisierung.

Sie müssen hohe Transparenz wahren, um sicherzustellen, dass Tools, die für das kontinuierliche Testing verwendet werden, hierzu passen. Der Ansatz muss mit der Geschäftsstrategie des Unternehmens koordiniert sein, und die IT muss Teil dieser Geschäftsstrategie sein, statt sie nur zu unterstützen.

Um dies zu erreichen, benötigt das Unternehmen fest zugeordnete Mitarbeiter. Es sollte nicht alles an einen Drittanbieter delegiert werden, aber Drittanbieter sollten auch nicht ausgeschlossen werden. Es ist akzeptabel, globale Systemintegratoren (GSIs) und globale Service Provider (GSPs) für die Zielerreichung heranzuziehen, aber das Unternehmen sollte auch eigene Mitarbeiter in den Prozess einbinden. Nutzen Sie Ihre eigenen Fachkenntnisse, aber untersuchen und verfestigen Sie auch Ihre Beziehungen mit Drittanbietern, da sie möglicherweise schon jetzt – oder in Kürze – über eine Lösung verfügen, die in die Pipeline integriert werden kann.

Identifizieren und bilden Sie spezialisierte Teams für die Hochskalierung der Plattformen, die die Anwendungspipeline und die wichtigen Messdaten für das kontinuierliche Testing unterstützen. Dies bedeutet, dass Sie eine vorhandene Organisation, der einige manuelle Tester und ein paar Techniker angehören, zu einer Organisation mit höherer Kapazität vergrößern.

Der Weg zum kontinuierlichen Testing bedeutet, dass der Umfang des Blackbox-Testing abnimmt, während das Whitebox-Testing zunimmt und wichtiger wird. Die Tester benötigen hierfür geeignete Kompetenzen. Daher muss ein CoE sich vom Center of Excellence zum Center of Enablement mausern. Dies wiederum verlangt weitere Veränderungen seitens der Teams und der Tester.

Big Data ist für das Testing unverzichtbar, vor allem, um die Codeabdeckung für alle Arten von Tests zu verfeinern. Der Markt für die Abdeckung der Regressionssuite wächst zurzeit explosionsartig, vor allem im Hinblick auf die Entwicklung einer theoretischen Methode, mit der ermittelt werden kann, ob unsere Regressionstests eine geeignete Codeabdeckung erreichen.

Unternehmen müssen auf Veränderungen vorbereitet sein und sie akzeptieren. Einige der wichtigen Meilensteine sind hier folgende:

DIE KI SPIELT EINE GRÖSSERE ROLLE

- Die Automatisierung der Automatisierung, d. h. die automatische Erzeugung von Tests bei Bedarf, wird Regressionssuites von Tests ersetzen, da diese nicht mehr aufrechterhalten werden können.
- Echtzeitbeobachtungen von Anwendern werden wichtiger. Daher werden Tests erforderlich, mit denen diese Beobachtungen überprüft werden können.
- Die Aufzeichnung und Wiedergabe gehört der Vergangenheit an.
- Außerdem wird der Schwerpunkt kontinuierlich stärker darauf gelegt, Anforderungen korrekt zu definieren sowie sie auf neue Weisen zu analysieren und zu identifizieren, was fehlt.

Das Führungsteam muss erkennen, wie schnell sich seine Umgebung verändert. Es wird zunehmend mit mobilen Technologien und Cloud-basierten Anwendungen gearbeitet, und Geschwindigkeit ist essenziell. Alles verändert sich in Richtung einer nahtlosen End-to-End Customer Experience. Cloud, DevOps, Agile, Big Data – das Endprodukt aus allen diesen Elementen ist eine nahtlose Customer Experience. Sie müssen entscheiden, wie Sie dies auf das Testing anwenden. An dieser Stelle begegnen sich die zwei Welten.

Der Anwender/Verbraucher muss den Wert wahrnehmen, den das Unternehmen beabsichtigt hat. Hier unterstützt das kontinuierliche Testing Innovationen.

Über das Internet of Things werden Geräte auf viele unterschiedliche Weisen vernetzt. Zurzeit stellt das kontinuierliche Testing eine Herausforderung für Browser und APIs dar. Im Hinblick auf physische Geräte, Mobile Devices und das Internet of Things hingegen werden Integrationszentralen, die ein kontinuierliches Testing ermöglichen, zur treibenden Kraft für Veränderungen. Wenn dies fehlt, wird das nicht unbemerkt bleiben.

Sie müssen immer drei Schritte vorausdenken und etwa vorausschauende Analysen nutzen.

Hervorragende QA- und QE-Mitarbeiter werden sich schnell weiterentwickeln. Ihre Kompetenzen werden zunehmen, und die für das kontinuierliche Testing notwendigen Fähigkeiten werden morgen ganz anders sein als heute. Unternehmen müssen sich vor allem auf Mobil- und Omnichannel-Lösungen konzentrieren.

Die flexibelsten Unternehmen können mithilfe einer neuen Unternehmenskultur des kontinuierlichen Testing nicht nur Marktanteile gewinnen, sondern auch die besten Mitarbeiter in diesem Bereich.

ZUSAMMENFASSUNG

Die Transformation zum kontinuierlichen Testing ist ein Weg, genau wie Agile und DevOps. Unternehmen und ihre IT-Teams werden auch Rückschläge erleben. Diese müssen sie erwarten und akzeptieren. Daher ist eine gute Planung bei jedem Schritt auf dem Weg zum kontinuierlichen Testing unerlässlich.

Das kontinuierliche Testing ist ebenso sehr eine Unternehmenskultur wie eine Technik. Mit einem entsprechenden Ansatz kann das gesamte Team aus Fehlern und Rückschlägen lernen. Dies ist ein großer Fortschritt gegenüber den voneinander getrennten Silos der Vergangenheit. Alle hervorragenden Entwicklungen und persönlichen Erfolge basieren auf einer festen Grundlage, in der sich Wissen mit Erfahrung verbindet.

Auch wenn das kontinuierliche Testing technologische Tools einsetzt, liegt sein Mittelpunkt in den Personen. Sobald sie mühelos und souverän arbeiten können, bringen sie die notwendige Energie, Vision und praktische Kompetenz hervor.

Aus diesen Erfolgen baut sich Schwung auf, und dieser wird die Unternehmen erfolgreich durch dieses neue weltweite Wirtschaftsumfeld tragen.

CS200-383435_0818