Qualitätssicherung

Kenntnisse über den Zweck von Code-Reviews

Agile Teams organisieren sich dank teamübergreifender Kompetenzen selbst. Dies wird teilweise durch Code-Reviews erreicht. Mit einem Code-Review lernen Entwickler nicht nur die Codebasis kennen, sondern erlernen auch neue Technologien und Verfahrensweisen, die ihre Kompetenzen erweitern.

Was genau ist also ein Code-Review:

Wenn ein Entwickler die Arbeit an einem Vorgang beendet hat, sieht sich ein anderer Entwickler den Code an und stellt Fragen wie:

- Gibt es offensichtliche Logikfehler im Code?
- Wurden alle anforderungsbezogenen Fälle implementiert?
- Reichen die neuen automatisierten Tests für den neuen Code aus?
- Müssen vorhandene automatisierte Tests neu geschrieben werden, um Änderungen im Code zu berücksichtigen?
- Entspricht der neue Code vorhandenen Formatierungsrichtlinien?

Code-Reviews sollten in den vorhandenen Prozess eines Teams integriert werden. Wenn ein Team beispielsweise Task-Branching-Workflows verwendet, solltest du einen Code-Review initiieren, nachdem der gesamte Code geschrieben wurde und automatisierte Tests ausgeführt und bestanden wurden, aber bevor der Code in den Upstream gemerkt wird. Damit wird sichergestellt, dass der Code-Reviewer den Code auf Probleme überprüft, die Maschinen übersehen. Außerdem wird verhindert, dass sich schlechte Entscheidungen bezüglich des Codes auf die Hauptentwicklung auswirken.

Was bedeutet das für ein agiles Team:

Jedes Team kann unabhängig von den Entwicklungsmethoden von Code-Reviews profitieren. Agile Teams profitieren jedoch besonders, da die Arbeit dezentralisiert über das gesamte Team hinweg erfolgt. Es gibt nie nur eine einzige Person, die einen bestimmten Teil der Codebasis kennt. Einfach gesagt tragen Code-Reviews dazu bei, Wissen einfacher über die Codebasis und das Team hinweg zu teilen.
Code-Reviews ermöglichen das Teilen von Wissen.

Agile Team sind unschlagbar flexibel: Sie können Aufgaben aus dem Backlog nehmen und von allen Teammitgliedern bearbeiten lassen. Teams können deshalb neue Aufgaben gemeinsam angehen, da niemand einen Engpass darstellt. Full-Stack-Entwickler können sowohl Front-end- als auch serverseitige Arbeiten erledigen.

Fachbegriff Schreibtischtest

Der Schreibtischtest ist ein Verfahren, das im Bereich der Softwareentwicklung verwendet wird, um Algorithmen oder Routinen auf Richtigkeit zu prüfen. Der Schreibtischtest wird nicht mit Hilfe eines Rechners durchgeführt, sondern vielmehr im Kopf des Entwicklers. Dazu werden für einen deterministischen und terminierenden Programmablauf eine Eingabe- und eine mögliche Ausgabemenge festgelegt. Anschließend wird mit jedem Element der Eingabemenge durch schrittweises Durchrechnen die Korrektheit des Programmablaufs

überprüft. Der gewählte Programmablauf verhält sich für das gewählte Eingabeelement genau dann korrekt, wenn das ausgegebene Element, also das Ergebnis, Teil der Ausgabemenge ist.

Die Tatsache, dass die Eingabemenge für einen Algorithmus beliebig groß sein kann, z. B. eine Untermenge der natürlichen Zahlen, erfordert vom Entwickler zur Überprüfung der Korrektheit des zu testenden Programms die schrittweise Durchrechnung mit allen Elementen der Eingabemenge. Gibt es für die Eingabemenge keine obere oder untere Schranke, dann kann der Schreibtischtest höchstens mit einigen plausiblen Eingaben ein Indiz dafür liefern, dass der Algorithmus richtig implementiert wurde.

Voraussetzungen für den Schreibtischtest sind also ein deterministischer und terminierender Programmablauf, die Kenntnis über plausible und sinnvolle Eingabeelemente, und die Kenntnis der zu den Eingabeelementen passenden Ausgabeelemente, um aus der jeweiligen Ausgabe auf die Richtigkeit schließen zu können. Das Ergebnis für die jeweilige Eingabe muss also bekannt sein.

Kenntnisse über Black-Box-Test, White-Box-Test, wesentliche Unterschiede

Black-Box-Test bezeichnet eine Methode des Softwaretests. Hierbei werden Tests anhand der Spezifikation/Anforderung entwickelt. Dies bedeutet, dass Tests ohne Kenntnisse über die innere Funktionsweise/Implementierung des zu testenden Systems entwickelt werden. Das zu testende Programm wird also als Black Box behandelt. Nur nach außen sichtbares Verhalten fließt in den Test ein. Im Gegensatz dazu werden White-Box-Tests mit Blick auf den implementierten Algorithmus entwickelt.

Zielsetzung:

Ziel ist es, die Übereinstimmung eines Softwaresystems mit seiner Spezifikation zu überprüfen. Ausgehend von formalen oder informalen Spezifikationen werden Testfälle erarbeitet, die sicherstellen, dass der geforderte Funktionsumfang eingehalten wird. Das zu testende System wird dabei als Ganzes betrachtet, nur sein Außenverhalten wird bei der Bewertung der Testergebnisse herangezogen.

Testfälle aus einer informalen Spezifikation abzuleiten, ist aufwändig und je nach Präzisionsgrad der Spezifikation u. U. nicht möglich. Oft ist daher ein vollständiger Black-Box-Test ebenso wenig wirtschaftlich wie ein vollständiger White-Box-Test.

Auch ist ein erfolgreicher Black-Box-Test keine Garantie für die Fehlerfreiheit der Software, da in frühen Phasen des Softwareentwurfs erstellte Spezifikationen spätere Detail- und Implementationsentscheidungen nicht abdecken.

Black-Box-Tests verhindern, dass Programmierer Tests „um ihre eigenen Fehler herum" entwickeln und somit Lücken in der Implementierung übersehen.
Ein Entwickler, der Kenntnisse über die innere Funktionsweise eines Systems besitzt, könnte unabsichtlich durch gewisse zusätzliche Annahmen, die außerhalb der Spezifikation liegen, einige Dinge in den

Die Vorteile von Black-Box-Tests gegenüber White-Box-Tests:

- Bessere Verifikation des Gesamtsystems.
- Testen von bedeutungsmäßigen Eigenschaften bei geeigneter Spezifikation.
- Portabilität von systematisch erstellten Testsequenzen auf plattformunabhängige Implementierungen.

Die Nachteile von Black-Box-Tests gegenüber White-Box-Tests:

- Zusätzlich eingefügte Funktionen bei der Implementierung werden nur durch Zufall getestet.
- Testsequenzen einer unzureichenden Spezifikation sind unbrauchbar.

Zudem sei genannt, dass die Unterscheidung Black-Box-Test vs. White-Box-Test teilweise von der Perspektive abhängt.
Das Testen einer Teilkomponente ist aus Sicht des Gesamtsystems ein White-Box-Test, da für das Gesamtsystem aus der Außenperspektive keine Kenntnisse über den Systemaufbau und damit die vorhandenen Teilkomponenten vorliegen.
Aus Sicht der Teilkomponente wiederum kann derselbe Test unter Umständen als Black-Box-Test betrachtet werden, wenn er ohne Kenntnisse über die Interna der Teilkomponente entwickelt und durchgeführt wird.

Kenntnisse über wichtige Qualitätsmerkmale der Softwarefunktionalität

Der Begriff der Softwarequalität spielt sowohl für die Entwicklung als auch für die Akzeptanz einer Software eine wichtige Rolle. Zur Bewertung der Softwarequalität werden verschiedene Merkmale bewertet und herangezogen.

Letztlich kann Qualität also mit dem Begriff der Güte eines Objekts oder Produkts definiert werden. Der Begriff der Softwarequalität bedeutet also die Güte einer Software. Um diese zu ermitteln, müssen allerdings Kennzahlen, Bereiche und Merkmale definiert werden, die eine Bewertung erlauben.

Wichtige Merkmale der Softwarequalität im Überblick

Entsprechend der oben genannten Definitionen spielen verschiedene Merkmale eine Rolle, wenn die Softwarequalität bewertet werden soll. Betrachten wir die verschiedenen Merkmale einmal genauer:

Funktionalität:
Der Funktionsumfang der Software ist ein wichtiger Punkt der Softwarequalität. Eine hochwertige Software muss in der Lage sein, die im Vorfeld definierten Leistungen zu erbringen und die gewünschten Funktionen zu erfüllen.

Zuverlässigkeit:
Auch die Zuverlässigkeit ist ein wichtiges Kriterium der Softwarequalität. Eine Software muss ein vorher definiertes Leistungsniveau über einen vorher definierten Zeitraum unter bestimmten Bedingungen halten können, um als zuverlässig zu gelten. Zudem müssen die bestehenden Funktionen der Software immer identisch funktionieren und ein immer gleiches Ergebnis liefern. Zur Bewertung der Zuverlässigkeit werden in vielen Unternehmen sogenannte Unit-Tests genutzt, welche genau diese Szenarien durchspielen und somit die Softwarequalität sicherstellen.

Effizienz:
Der Begriff der Effizienz ist eine Frage der Performance. Das bedeutet, dass eine hohe Softwarequalität die Lauffähigkeit und Arbeitsleistung der Software unter bestimmten, vorher definierten Anwendungsumgebungen und Hardwarestrukturen bewertet. Je effizienter die Software arbeitet, umso größer die Bandbreite an möglicher Hardwareunterstützung. Somit steigt durch eine gute Effizienz auch die Bandbreite der möglichen Nutzer.

Benutzbarkeit:
Die Frage nach der Benutzerfreundlichkeit der Software umfasst sowohl den Aufwand, welcher zum Erlernen einer Software notwendig ist, als auch die Leistung der Software in den verschiedensten Nutzungsszenarien. Die Benutzbarkeit wird unter anderem durch Software-Tester, aber auch durch die Reaktionen und das Feedback der Kunden ermittelt. Eine Software mit einer guten Benutzbarkeit lässt sich beispielsweise möglichst intuitiv bedienen.

Übertragbarkeit:
Unter dem Begriff der Übertragbarkeit wird die Kompatibilität zu anderen Betriebssystemen beschrieben. Es wird die Frage gestellt, ob und mit welchem Aufwand eine Software in ein anderes System migriert und dort lauffähig gemacht werden kann. Hierunter fallen nicht nur unterschiedliche Betriebssysteme, sondern unter anderem auch unterschiedliche Anwendungsumgebungen wie der mobile Bereich. Je flexibler die Software ist und je einfacher die Portierung in ein anderes System, umso höher wird die Softwarequalität bewertet.

Änderbarkeit:
Der Begriff der Änderbarkeit misst und bewertet die Möglichkeit, die Software durch Programmierung zu verändern. Je modularer und klarer strukturiert eine Software aufgebaut ist, umso größer ist deren Änderbarkeit und umso höher die Bewertung der Softwarequalität.

Probleme der einzelnen Merkmale

Es gibt allerdings bei der Verbesserung der Softwarequalität Probleme. Denn es ist in der Regel nicht möglich, alle Qualitätseigenschaften gleichermaßen zu verbessern, da es unter anderem Zielkonflikte der unterschiedlichen Eigenschaften gibt. Dementsprechend ist es immer notwendig, bei der Qualitätssicherung einer Software-Schwerpunkte zu setzen und diese zu priorisieren.

Kenntnisse über Changemanagement

Veränderungen lassen sich nicht aufhalten, aber sie lassen sich erfolgreich managen.

Genau darum geht es im Change-Management: Prozesse, Strukturen, Systeme, Strategien, Werte und Verhaltensweisen eines Unternehmens bzw. einer Organisation werden auf die neuen, sich ändernde Marktbedingungen angepasst und ausgerichtet.

Change-Management bezeichnet fortlaufende Veränderungen der Strukturen einer Organisation zu den Anpassungen an sich wandelnde Umwelt- und Rahmenbedingungen.

Change-Management ist umso notwendiger, je schneller sich die Umwelten von Unternehmen ändern.
Besondere Dynamik entsteht durch Faktoren wie:

- Internationalisierung
- Globalisierung von Absatz- und Finanzmärkten
- rasanter technische Fortschritt
- politische Veränderungen, z.B. gesetzliche Auflagen
- soziale Veränderungen, wie demographischer Wandel, höhere Frauenerwerbstätigkeit
- größeres Bedürfnis nach Work-Life Balance bei Arbeitnehmern
- ökologische Veränderungen, wie Verknappung von Ressourcen

Die Auswirkungen dieser Dynamik sind allgegenwärtig: Fusionen, Pleiten, Neugründungen, Entlassungen und zunehmend geforderte und belastete Führungskräfte gehören zum Alltagsgeschehen. Unternehmerisches Wirtschaften und betriebliches Management vollziehen sich heute unter ganz anderen Voraussetzungen als noch vor wenigen Jahren.

Mitverantwortlich dafür sind diese fünf Rahmenbedingungen, die heute weitgehend über Erfolg oder Misserfolg unternehmerischen Wirtschaftens entscheiden:

- Innovationssprünge in der Informatik und Telekommunikation
- Verknappung der Ressource Zeit
- Verknappung der Ressource Geld
- Dramatische Steigerung der Komplexität
- Interkulturelle Zusammenarbeit in einer globalen Ökonomie

Bewährte Instrumente des Change-Managements sind:

Diagnose der Defizite und Probleme: Was ist der Fall, was ist Auslöser?
Methoden: Dokumenten-Analyse, Selbstbewertung, Konkurrenzanalyse, Marktbewertung

Analyse: Welche Probleme haben wir?
Methoden: Organisationsanalyse, Prozessanalyse, Arbeitsanalyse

Zielformulierung und Konzeption: Was wollen wir? Was ist unser Ziel?
Methoden: Kreativitätstechniken, Moderation, Kommunikationsmanagement, Konzepte, Zeitrahmen, Budget.

Umsetzung der Konzeption: Wie erreichen wir unser Ziel?
Methoden: Prozessberatung und -Management, Projektmanagement, Kommunikationsmanagement, Workshop, Individual- und Teamcoaching, Großgruppenmoderation, Inhouse-Beratung

Kontrolle und Bewertung der Ergebnisse: sind gesetzte Ziele erreicht worden?
Methoden: Mitarbeiterbefragung, Interviews, Bewertungskatalog, Qualitätszirke

Fachbegriff Versionierung und deren Nutzen

Versionierung ist ein systematischer Ansatz zum Management aller Änderungen an einem Produkt oder System. Anwender wollen damit sicherstellen, dass keine unnötigen Änderungen vorgenommen werden, alle Änderungen dokumentiert, Dienste nicht unnötig gestört und Ressourcen effizient eingesetzt werden. In der Informationstechnologie (IT) ist Versionsverwaltung ein Bestandteil der Änderungskontrolle.

Der Prozess für die Versionskontrolle besteht normalerweise aus einer festen Abfolge von Schritten, die durch das Einreichen einer Änderungsanfrage ausgelöst werden. Typische IT-Änderungsanfragen umfassen das Hinzufügen von Funktionen zu Softwareanwendungen sowie die Installation von Patches und Upgrades von Netzwerkgeräten oder -systemen.

Typische Prozesse für die Versionsverwaltung:
Hier ist ein Beispiel für einen sechsstufigen Prozess für eine Softwareänderungsanfrage:

- Dokumentieren der Änderungsanforderung: Die Änderungsanfrage oder der Vorschlag des Kunden wird kategorisiert und zusammen mit informellen Bewertungen

der Bedeutung dieser Änderung und der Schwierigkeit ihrer Umsetzung aufgezeichnet.
- Formale Bewertung: In diesem Schritt bewerten Mitarbeiter die Gründe für die Änderung sowie deren Risiken und Vorteile. Geht die Änderung durch, wandert sie als nächstes zum Entwicklerteam. Wird die Änderungsanfrage abgelehnt, dokumentieren die Mitarbeiter dies und antworten dem Kunden.
- Planung: Das für die Änderung verantwortliche Team erstellt einen detaillierten Plan für deren Design und Implementierung sowie für das Zurücksetzen der Änderung, falls sie misslingt.
- Entwerfen und Testen: Das Team entwirft das Programm für die Softwareänderung und testet es. Passiert die Änderung die Tests, fordert das Team die Genehmigung und ein Implementierungsdatum an.
- Implementieren und Überprüfen: Das Team implementiert das Programm und die beteiligten Personen überprüfen die Änderung.
- Abschließende Bewertung: Ist der Kunde mit der Umsetzung der Änderung zufrieden, schließen die Mitarbeiter die Änderungsanfrage. Ansonsten bewerten sie das Projekt erneut und wiederholen die Schritte.

Kenntnisse über Problemmanagement

Problem-Management umfasst alle Prozesse und Aktivitäten, die für die Verwaltung des Lebenszyklus von Problemen im IT-Service nötig sind. Hauptziel ist es, Probleme und die daraus resultierenden Vorfälle zu vermeiden. Bei bereits aufgetretenen Vorfällen versucht das Problem-Management zu verhindern, dass sie sich wiederholen. Bei unvermeidlichen Problemen sollen die Auswirkungen auf das Unternehmen minimiert werden.

Um das Problem-Management zu verstehen, muss man definieren, was ein Problem ist. ITIL definiert ein Problem als die Ursache für einen oder mehreren Vorfällen. Ein Problem ist eine zugrunde liegende Bedingung, die negative Auswirkungen auf den Service haben könnte und daher behandelt werden muss.

Probleme haben einen spezifischen Lebenszyklus:

- Entstehung des Problems (oft durch eine Veränderung in der Umgebung).
- Identifizierung und Phasen der Diagnose und Behebung.
- Lösung des Problems entweder durch eine Maßnahme oder durch das Wegfallen der zugrunde liegenden Situation.

Problem-Management ist zum einen ein transaktionaler Prozess, in dem der Lebenszyklus eines einzelnen Problems verwaltet wird. Dazu gehört die sogenannte Root Cause Analysis (RCA), um die Ursache von Vorfällen zu diagnostizieren und die geeigneten Lösungsschritte festzulegen. Es ist auch dafür verantwortlich, dass alle Lösungen sicher und effektiv in Übereinstimmung mit den Richtlinien und Verfahren des Change- und Release-Managements umgesetzt werden.

Problem-Management umfasst aber auch Portfolio-Management:
Es werden Entscheidungen getroffen, welche Probleme behandelt werden sollen, welche Ressourcen dafür eingesetzt werden und welche Risiken diese Probleme für das Unternehmen darstellen. Der Portfolioteil des Problem-Managements ist für die Pflege von Informationen über bestehende Probleme, entwickelte Workarounds und die identifizierten Lösungsoptionen zuständig. Diese Informationen ermöglichen es Führungskräften, Entscheidungen zu treffen, die die Anzahl und die Auswirkungen von Vorfällen reduzieren.