Die Welt der Software-Qualität

Software-Qualität ist nicht nur Testen und Qualitätssicherung, sondern viele Themen sind in diesem Zusammenhang wichtig wie z.B. Requirements-Engineering, Change-Management, Code-Qualität, Risiko-Einschätzungen, angemessene Prozesse, Automatisierung, Vorgehensweisen und Prozesse, Tools und Tool-Integration, etc. In diesem Beitrag erhalten Sie einen Überblick über wichtige Themen aus Sicht der Software-Qualität.

Wie die Erkenntnis wächst!

Es gibt drei Arten von Organisationen:

- Die einen bewirken, dass etwas geschieht,
- die anderen beobachten, dass etwas geschieht und
- wieder andere fragen sich, was geschehen ist!

In der Welt der Software-Qualität ist die erstgenannte Gruppe leider noch in der Minderheit. Jedoch hat es in den letzten 15 Jahren eine deutliche Verschiebung hin zu den engagierten Organisationen gegeben.

Immer mehr Ausbildungsinstitutionen (UNIs, FHs, etc.) nehmen Software-Qualitätsthemen in das Ausbildungsprogramm und die Absolventen tragen dieses Wissen hinaus in die Unternehmen.

Wo früher in den Software-Organisationen nur Entwickler gearbeitet haben, finden wir heute auch schon einige Tester. Wo früher die Tester diejenigen waren, die ins Testen „abgeschoben“ wurden, weil sie nicht gut genug programmieren konnten, sind die Tester heute oft höher qualifiziert als die Entwickler und in einer Schlüsselrolle zwischen Produktmanagement, Kunden und Entwicklung.

Viele Software-Organisationen machen sich auch Gedanken über die Prozesse und (weniger oft) werden diese auch systematisch definiert und weiterentwickelt. Der Trend geht in die richtige Richtung!

Die Ausrichtung auf Software-Qualität bedeutet Veränderung

Viele Entwicklungsorganisationen verharren in Selbst-zufriedenheit mit folgendem Denken und Handeln:

- Wir sind doch der Marktführer/ die Besten
- Dazu haben wir jetzt keine Zeit
- Das ist nur eine Konjunkturkrise
- Bei uns ist das alles ganz anders
- Wir machen doch Gewinne, unser Geschäft läuft doch
- Den anderen geht es auch nicht besser
- Wir schwärmen von Erfolgen der Vergangenheit
- Wir fokussieren uns auf das Tagesgeschäft
- Mit unseren Mitarbeitern geht das nicht
- Etc.

Organisationen, die so oder ähnlich denken, werden sich aus dem Sumpf der Verweigerung, Verwirrung und Selbstzufriedenheit nicht lösen und typischerweise von anderen Organisationen überholt und in den Schatten gestellt.

Leider gilt auch oft das Sprichwort „Nachdem wir das Ziel aus den Augen verloren hatten, verdoppelten wir unsere Anstrengungen!“

Was können Sie nun tun, um sich mit Ihrer Organisation weiterzuentwickeln?

- Stellen Sie die Denkgewohnheiten in Frage und schauen Sie gemeinsam über den Tellerrand.
- Erzeugen Sie Aufbruchstimmung und geben Sie dem Denken eine neue Richtung z.B. durch eine mit dem Management entwickelte gemeinsame Vision.

- Brechen Sie eingefahrene Gruppenstrukturen auf und bewegen Sie die Mitarbeiter zu Veränderungen - ev. auch physisch, z.B. Job-Rotation zwischen Entwicklern und Testern oder gemischte Teams wie in der agilen Entwicklung.

- Leben Sie Veränderungen selbst vor, auch wenn es vorerst nur in einem eingeschränkten Organisations- oder Themenbereich ist.
- Wichtig ist, dass alle beteiligten Personen inkl. dem verantwortlichen Management anfangen, über Veränderung nachzudenken und begreifen, dass Veränderungen notwendig sind!

In diesem Sinn kann der vorliegende Knowledge-Letter einige Denkanstöße liefern, welche Themen zur Verbesserung der Software-Qualität in der Organisation verändert und optimiert werden können.

Einleitung und Abgrenzung

Dieser Beitrag soll verschiedene Kernthemen, die für den Bereich Software Qualität wichtig sind, ansprechen und vor allem als Gedankenanstoß dienen, welche Fragen man sich selbst in diesen Themen stellen sollte und worauf man in der eigenen Organisation achten sollte.

Dieser Beitrag soll NICHT eine vollständige Abhandlung zum Thema Software Qualität sein und es darf aufgrund des geringen Umfangs auch nicht erwartet werden, dass umfangreiche Analysen und Lösungsansätze vorgestellt werden.

Im Rahmen dieses Knowledge-Letters wurde auf die Themen Requirements-Engineering, Testen und Testautomatisierung etwas genauer eingegangen. Es gibt eine Reihe von weiteren wichtigen Themen, die großen Einfluss auf die Software-Qualität haben und die nachfolgend angeführt sind. Um den Rahmen dieses Knowledge-Letters nicht zu sprengen, werden diese Themen in diesem Knowledge-Letter nicht umfangreicher behandelt, sondern es werden nur einige interessante und wichtige Fragestellungen im Zusammenhang mit diesen Themen angesprochen. In weiteren Knowledge-Lettern werden wir diese wichtigen Themen dann noch gezielt behandeln.

Requirements

Das Thema Requirements ist ein Dauerbrenner in fast jeder Entwicklungsorganisation und das schon seit Jahrzehnten. Die Langzeit-Beobachtung lässt fast befürchten, dass viele Organisationen dieses Thema auch in den nächsten Jahrzehnten noch nicht nachhaltig gelöst haben werden.

Viele Betroffene tappen hier noch ziemlich im Dunkeln - nicht nur, was die Inhalte der Requirements an sich betrifft, sondern es sind hier einfach in den meisten Fällen noch viele Fragestellungen offen wie z.B.

- **Umfang, Detailliertheit und Qualität der Requirements:** Einerseits ist dies unter dem Aspekt zu sehen, dass möglichst überflüssiger Ballast und administrativer Aufwand vermieden wird. Andererseits jedoch soll auch der Wissenstransfer durch den Prozess sichergestellt werden und die Requirements sollen als gute Basis für die weiteren Prozess-Schritte und die Dokumentation des Systems auch eine entsprechend gute Qualität aufweisen.
- **Unterscheidung WAS – WIE:** Oft werden Lösungsausarbeitungen (das WIE es umgesetzt wird) und die Anforderungen aus der Kundensicht (das WAS der Kunde gerne haben möchte) vermischt.

  In den heute üblichen iterativen Vorgehensweisen ist es auch notwendig, zwischen dem WAS und dem WIE einen ständigen Abgleich und Konsens zu schaffen. Jedoch sollte schon aus dem Aspekt der unterschiedlichen Verantwortlichkeit und Kompetenz für verschiedene Themen darauf geachtet werden, dass diese beiden Sichten nicht vermischt werden.

  Aus einer Anforderung sollte klar hervorgehen, ob sie in der Kundenverantwortung liegt oder ob sie ein Vorschlag des Umsetzers ist.
- **Aufbau und Struktur:** Die Strukturierung der Requirements ist auch oft ein Thema. Viel zu schnell begibt man sich dabei in die Tiefen des zu spezifizierenden Systems und vergeudet wertvolle Zeit mit Details, die in der jeweiligen Phase noch gar nicht relevant sind. Eine angemessene Top-Down-Vorgehensweise von der Idee über Projektziele, Kontextdarstellungen, Business-Prozesse, Use-Cases, Features bis hin zur detaillierten Masken- und Schnittstellenbeschreibung ist hier meist zielführend, um den Überblick nicht zu verlieren.
- **Genügend Zeit? Der Chef bzw. Auftraggeber drängt?** In vielen Unternehmen ist es ein Bewusstseins-Thema nach dem Motto „Wieso wird denn hier so viel geschrieben? Wäre es nicht besser, wenn endlich gearbeitet wird?“. In solchen Organisationen erübrigt sich das Thema Requirements-Spezifikation meist von selbst.

  In vielen Fällen ist es aber auch der Zeitdruck, der meist aus einer falschen Aufwandsschätzung und Planung resultiert. Es wird dann einfach zu wenig Zeit für eine gute Spezifikation eingeplant oder auch bei guter Planung kann es dem Auftraggeber oder Chef oft nicht schnell genug gehen, bis er endlich etwas Vorzeigbares sieht. Meist passiert dies dann auf Kosten der Qualität im Prozess.
- **Requirements-Engineering in agilen Projekten:** Aufgrund der Verbreitung der agilen Vorgehensweisen ist dies

immer mehr ein wichtiges Thema, da die meisten agilen Vorgehen darauf leider nicht genau eingehen.

User-Stories und andere agile Requirements-Artefakte werden zwar erwähnt, aber leider nur sehr oberflächlich beschrieben. Daher sollte sich jede Organisation, die agile Vorgehensweisen anwendet, auch Gedanken darüber machen, wie mit den Requirements umgegangen wird.

Sehr oft beschränkt man sich leider nur auf 1-2 Ebenen wie z.B. Epics und User-Stories. Themen wie die Business-Prozess-Sicht, Architekturdokumentation, nicht-funktionale Anforderungen und andere Requirements-Themen sollten jedenfalls auch berücksichtigt werden.

- **Prüfung der Requirements - Test-Driven:** Testgetriebenes Vorgehen ist gerade in Verbindung mit der Requirements-Spezifikation eine sehr gute Methode, um eine gute Qualität der Requirements sicherzustellen.

  Die Tester sollen bei jedem Requirement (User-Story, Feature, User-Interface, Schnittstelle, etc.) nach dessen Erstellung oder Änderung einen Review hinsichtlich Testbarkeit durchführen. Jedes Requirement sollte durch die Tester zur Implementierung mit freigegeben werden müssen. Bevor nicht der Tester weiß, wie er das dann testen kann und sein OK erteilt, darf eine Anforderung nicht in die Feature-Liste oder Backlog für die Implementierung aufgenommen werden.

- **Passendes Tool:** Das meistverwendete Tool im Requirements-Engineering ist nach wie vor die Textverarbeitung oder Tabellenkalkulation. Diese sind zwar flexibel bei der Erstellung von Anforderungen, jedoch ziemlich unbrauchbar, wenn es um Änderungsverfolgung, Versionierung, Attribution und andere Themen im Requirements-Management geht. Mittlerweile haben auch viele Issue– bzw. Bug-Tracking-Werkzeuge Einzug ins Requirements-Management gehalten. Hier werden zwar ein paar Punkte besser gelöst wie in der Textverarbeitung oder Tabellenkalkulation, jedoch sind diese Tools insgesamt nicht geeignet, größere Mengen an Requirements gut zu verwalten. Diese Tools sind gut dazu geeignet, Anfragen und Aufgaben und deren Abarbeitung zu steuern und nicht dazu, um strukturiert Inhalte zu sammeln und darzustellen.

  In vielen Fällen führt die Verwendung von Issue-Tracking-Tools im Requirements-Engineering dazu, dass die beteiligten Personen völlig den Überblick über die Anforderungen verlieren, weil in diesen Tools Zusammenhänge und strukturierte Hierarchien zwischen Requirements und anderen Artefakten meist nicht brauchbar dargestellt werden können.

  Es kann daher nur abgeraten werden, Textverarbeitung, Tabellenkalkulation oder Issue-Tracking-Tools für das Requirements-Management zu verwenden.

  Es muss klar getrennt werden zwischen der inhaltlichen Darstellung und Strukturierung von Anforderungen, die idealerweise durch geeignete Requirements-Management-Tools erfolgt, und der Steuerung der Umsetzung dieser Requirements, wofür sich dann Tracking-Tools wieder recht gut eignen.

Testen

Das Testen an sich muss heute kaum mehr argumentiert werden. Es gibt aber auch hier einige Themen, die immer wieder hinterfragt werden sollten:

- **Testen ohne Grundlage:** Es stellt sich häufig die Frage, was die passende Testgrundlage ist. Testen die Tester das, was die Entwickler ihnen geben bzw. sagen? Testen die Tester nach den Vorgaben, die ihnen die Entwickler geben?

  Oder haben die Tester angemessene Anforderungsspezifikationen, Qualitätsrichtlinien, Definitions of Done, vordefinierte Guidelines und Checklisten etc., an denen sie sich bei der Testfallerstellung und Durchführung orientieren können?

  Ein Software-Tester ohne Testgrundlage ist wie ein Wanderer ohne Karte und Ziel. Er wird herumirren und ev. zufällig oder mit seinem Hausverstand und Fachwissen den einen oder anderen „Fehlerberg“ finden, insgesamt jedoch meist ziemlich ineffizient arbeiten.

- **Vorgehensweise beim Testen:** Leider haben sich die Auftraggeber und Umsetzungsverantwortlichen vor Beginn des Entwicklungsvorhabens oft nicht überlegt, wie das Testen überhaupt durchgeführt werden soll.

  Gibt es eine Test-Strategie und ein Testkonzept, in dem die wichtigsten Themen für die Testdurchführung wie z.B. Testziele, Test-Endekriterien, Test-Abdeckung, Testorganisation, Testumgebung, usw. beschrieben sind und die den Testern dann als Leitfaden für die Testdurchführung dienen? Gibt es einen Prozess, in dem die Vorgehensweise des Testens beschrieben ist und der allen Beteiligten bekannt ist?

  Ist der Sinn des Testens, dass die Tester überprüfen, ob die Entwickler die Anforderungen des Kunden auch

richtig umgesetzt haben? Oder ist der Tester ein Unwissender im Team, der sich mühsam die benötigten Informationen zusammenkratzen muss und nichts zu sagen hat?

- **Verschiedene Testebenen:** Manche Unternehmen betreiben Unit-Tests in der Entwicklung schon recht umfangreich und sind dann der Ansicht, dass das Testen damit gut erledigt ist.

  Das ist leider ein Irrtum, da mit den Unit-Tests nur ein bestimmter Anteil der benötigten Tests abgedeckt werden kann. Abhängig von der Art des entwickelten Systems sind das z.B. zwischen 30 und 60% der benötigten Tests.

  Je nach Komplexität des Systems sind jedenfalls auch Integrationstests (ev. sogar mehrere Ebenen wie Integrationstest zwischen internen Komponenten und Integrationstest in der Kundenumgebung) und Systemtests durchzuführen, die typischerweise andere Sichtweisen wie die Unit-Tests abdecken und daher nicht einfach weggelassen werden können.

- **Rolle des Testers:** Was hat der Tester eigentlich in der (Entwicklungs-)Organisation zu sagen? Welche Verantwortung trägt er und welche Befugnisse hat er?

  Wenn der Tester nur als notwendiges Übel oder als „Feigenblatt der Qualität“ in der Entwicklung betrachtet wird, dann läuft etwas falsch.

  Der Tester bzw. Testmanager sollte entsprechend seiner Verantwortung auch angemessene Befugnisse haben wie z.B.

  - Festlegen, welche Bugs in einem Sprint/Release noch vor Freigabe ausgebessert werden müssen

  - Kategorie eines Bugs festlegen

  - Sprint mit freigeben

- **Zusammenarbeit mit Entwicklern:** Sind die Tester in die Entwicklung integriert oder sind die Tester eigenständig von den Entwicklern organisiert? Manchmal führen die Tester ein abgeschiedenes Dasein abseits der Entwicklung und die Entwicklung geht an ihnen im wahrsten Sinne des Wortes vorbei.

  In anderen Organisationen sind die Tester so in das Entwicklungs-Team integriert, dass sie schon fast nicht mehr wissen, dass es sich um Tester handelt.

  Die Tester sollten mit der Entwicklung eng zusammenarbeiten – schließlich sitzen ja alle im selben Boot und haben dasselbe Ziel, nämlich den Kunden durch gute Software zufrieden zu stellen.

  Die Tester sollten jedoch so eigenständig in ihrer Rolle und organisatorischen Position sein, dass Sie ihre Verantwortung und Aufgaben auch effektiv erfüllen können und das 4-Augen-Prinzip der Qualitätssicherung gewahrt bleibt.

- **Testgetriebene Spezifikation und Entwicklung:** Testgetriebene Vorgehensweise ist sowohl in der Spezifikation (siehe Abschnitt „Requirements“) als auch in der eigentlichen Umsetzung eine empfehlenswerte Vorgehensweise.

  Es gibt hier natürlich gewisse Abstufungen und „Härtegrade“ die man je nach Organisation und Umfeld für sich abstimmen muss.

  Die Aktualität und weite Verbreitung dieser Vorgehensweise zeigt sich auch mit den Ansätzen wie Behaviour-Driven-Development, welche im Wesentlichen Abwandlungen der testgetriebenen Vorgehensweise sind.

- **Testmanagement- und Reporting-Tools:** Ähnlich wie in der Anforderungsspezifikation zeigt sich auch im Testen, dass die Tabellenkalkulation oft noch das Tool der Wahl ist. Für eine überschaubare Anzahl von Testfällen mag dies auch gerechtfertigt sein. In größeren Projekten oder langfristigen Produktentwicklungen ist dies aber meist nicht angemessen und ineffizient.

  Im Testmanagement werden daher mittlerweile zur Verwaltung der Testfälle auch immer öfter professionelle Tools eingesetzt.

  Die gesamte Testplanung und -durchführung wird durch einen professionellen Tool-Einsatz effizienter.

Test-Automatisierung

- **Wann soll / kann automatisiert werden?** 100% Testautomatisierung ist nicht möglich, aber auch nicht sinnvoll! Manche Testfälle sind nur automatisiert umsetzbar wie z.B. das synchrone Auslösen von mehreren Events. Testautomatisierung wird jedoch in den meisten Fällen eine wirtschaftliche Entscheidung sein – es macht nur dann Sinn, einen Testfall zu automatisieren, wenn der Aufwand für die Automatisierung mit großer Wahrscheinlichkeit auch geringer sein wird als der Aufwand für die manuelle Testdurchführung (betrachtet über den Lebenszyklus des Testfalls).

- **Wie viel soll / kann automatisiert werden?** Man kann davon ausgehen, dass durch Unit-Tests und automatisierte Integrations- und Systemtests in typischen Software-Projekten ca. 80 % der Testfälle sinnvoll automatisiert werden können und über den Life-Cycle auch Kosteneinsparungen bringen. Bei 20% der Testfälle wird nach wie vor die manuelle Testdurchführung notwendig oder sinnvoll sein.

- **Welche Testfälle sollen automatisiert werden?** Chaos sollte man nicht automatisieren – das bringt nur schnelleres Chaos.

Da die Testautomaten (noch) nicht in der Lage sind, die fehlenden Teile einer schlechten Testspezifikation automatisch zu ersetzen und daraus gute automatisierte Testfälle zu machen, ist eine der wichtigsten Voraussetzungen, dass eine gute Testspezifikation für die zu automatisierenden Testfälle zugrunde liegt. Während für den fachlich gebildeten Menschen noch klar ist, was „ein gültiger Kunde ist einzugeben“ bedeutet, verzweifeln an einer solchen Formulierung die Automatisierungstools natürlich.

Wesentlich für die Auswahl der zu automatisierenden Testfälle ist natürlich die Ausführungshäufigkeit eines Testfalls. Die einfache Regel, die für die meisten Projekte passen wird, ist, dass Testfälle, die mit großer Wahrscheinlichkeit im Lebenszyklus mehr als 6-mal ohne Änderung durchgeführt werden, automatisiert werden sollten.

Sehr oft wird dies auch damit zusammenhängen, dass dies die risikoreichen oder die wichtigen Testfälle eines Systems sind. Also sind auch Einstufungen wie z.B. „Kerngeschäftsprozess“, „zentrale Funktion“ oder „hohes Risiko“ typische Indikatoren für zu automatisierende Testfälle.

- Ein weiterer Aspekt ist die Stabilität der zu automatisierenden Testfälle. Es ist einleuchtend, dass es wenig Sinn macht, einen Testfall zu automatisieren, der sich mit großer Wahrscheinlichkeit bei der nächsten Testdurchführung schon wieder geändert hat. Ein guter Wert ist hier eine Änderungs-Schwelle von 20% über 2 Iterationen anzusetzen — das heißt, wenn sich ein Testfall bzw. die dem Testfall zugrundeliegenden Softwareteile über zwei Iterationen nicht mehr als 20% verändern, dann ist das ein Indiz, dass dieser Testfall „reif“ für die Automatisierung ist. Hier sollte beachtet werden, dass dies primär für die Integrations- und System-Tests gilt, die außerhalb der Entwicklung automatisiert werden. Unit-Tests (zumindest die „Standard-Fälle“) sollten schon im Rahmen der jeweiligen Code-Entwicklung erstellt werden. Für die umfassende Programmierung der speziellen Unit-Test-Fälle sollte dann aus wirtschaftlichen Überlegungen ebenfalls die vorher genannte „Reife-Regel“ beachtet werden.
- **Unit-Test ist nicht alles! Alle Ebenen automatisieren!** Sehr häufig konzentrieren sich Software-Entwicklungsorganisationen, in denen das Testen primär aus der Entwicklung getrieben wird, auf das Thema Unit-Tests. Durch Unit-Tests alleine können aber nicht alle notwendigen Tests abgedeckt werden. Unit-Tests werden typischerweise ca. 40-50% der notwendigen Tests darstellen — bei Software mit wenig oder keiner Benutzeroberfläche ev. auch etwas mehr. Der Rest sind jedoch Integrations– und Systemtests, die ebenfalls betrachtet und durchgeführt werden müssen (z.B. systemübergreifende Schnittstellen oder komplexe Business-Prozesse) und die meist durch Unit-Tests nicht oder nur sehr umständlich gelöst werden können.
- **Ist Testautomatisierung durch Fachtester möglich?** Testautomatisierung war bislang eine Domäne der Entwickler. Es wurde Code erstellt, um anderen Code zu testen. Einige Testautomatisierungstools und Frameworks haben schon recht komfortable Management-Oberflächen und kapseln die technischen Aspekte der Automatisierung vor dem Benutzer.

  Dies soll auch Fachtestern, die keine Software-Entwickler sind, scheinbar sehr einfach ermöglichen, Tests für komplexe Systeme zu automatisieren.

  Dabei wird aber oft übersehen, dass es gewisser Voraussetzungen bedarf, damit dies funktioniert:

  - Die zugrundeliegende technische Komplexität des zu testenden Systems muss vorher in allen Aspekten gekapselt und abstrahiert werden. Dazu benötigt es nach wie vor einen erfahrenen Entwickler.
  - Die Fachtester müssen neben einem guten Test-Methodenwissen auch über ausreichendes Verständnis der technischen Zusammenhänge und der Architektur des zu testenden Systems verfügen.
  - Die Fachtester können meist auch nur einen Teil der Testfälle, welche die fachlichen Aspekte abdecken, erstellen.

  Wenn diese Voraussetzungen gegeben sind, können Fachtester die Testautomatisierung sicherlich sehr gut unterstützen.

  Es sollte jedoch zusätzlich zu den Fachtestern in der Testautomatisierung jedenfalls auch Software-technisch erfahrene und ausgebildete Testautomatisierer geben, die den Fachtester in dieser Hinsicht unterstützen und die primären technischen Teile der Automatisierung übernehmen und durchführen.

  Fachliche Tests, die zumeist die Systemtest-Sichtweise abdecken, sind auch nicht alles. Es sind nach wie vor zusätzlich Unit-Tests und komplexe Integrationstests zu erstellen, für welche sich Software-Entwickler besser eignen als der reine Fachtester.
- **Wartung / „Sterben“ der autom. Testfälle:** Im Rahmen der Testautomatisierung muss auch berücksichtigt werden, dass sich die zugrundeliegende Software ändert. Damit ist natürlich auch verbunden, dass die darauf aufbauenden Testfälle entsprechend angepasst werden müssen.

  Weiters muss auch regelmäßig überprüft werden, ob der einzelne Testfall überhaupt noch erforderlich ist bzw.

Sinn macht. Gegebenenfalls sind dann Testfälle auch zu eliminieren.

Die Wartung der automatisierten Testfälle kann mitunter beträchtlichen Aufwand verursachen. Daher wurde vorher die „Reife-Regel“ erwähnt, an der man sich orientieren sollte, um den Wartungsaufwand möglichst gering zu halten. Gänzlich wird sich der Wartungsaufwand nicht vermeiden lassen. Bei guter Testautomatisierung muss immerhin noch damit gerechnet werden, dass der Wartungsaufwand ca. 20% des Automatisierungs-Aufwands ausmacht.

- **Testautomatisierung – Resümee:** Testautomatisierung ist für eine effiziente Durchführung des Testens unumgänglich. Die Entwickler werden dadurch auch „mutiger“ und das laufende Refactoring eines Systems wird einfacher. Testautomatisierungs-Tools und Frameworks liefern eine gute Basis. Jedoch ist zu beachten, dass die Auswahl eines Testwerkzeugs allein noch keine Test-Strategie darstellt.

  Auch (Unit-Test-)Entwickler und Testautomatisierungs-Fachtester müssen entsprechend ausgebildet und unterstützt werden, damit Testautomatisierung effektiv und effizient durchgeführt werden kann.

Change-Management

Change-Management im Rahmen dieses Knowledge-Letters beschäftigt sich mit Änderungen an Dokumenten, Code und anderen Artefakten im Laufe der Software-Entwicklung. Nicht gemeint ist hier in diesem Artikel das Management organisationaler Veränderungen. Im Zusammenhang mit Change-Management sollte sich eine Entwicklungsorganisation unter anderem folgende Fragen stellen:

- Was ist überhaupt ein Change? Jede kleine Änderung oder nur bestimmte Arten von Änderungen?
- Wann im Entwicklungs-Prozess beginnt Change-Management?
- Für welche Arten von Dokumenten, Code, etc. macht es Sinn, Change-Management durchzuführen?
- Muss ein Change alle Phasen des Entwicklungsprozesses durchlaufen?
- Wie wird ein Change erkannt bzw. ausgelöst?
- Wie dokumentiert man einen Change am besten?
- Wer muss über einen Change informiert werden?
- Wann ist ein Change abgeschlossen und wie wird das festgestellt?
- Was sind Changes in agilen Projekten und wie geht man damit um — innerhalb und außerhalb einer Iteration?
- In welchem Zusammenhang steht Change-Management mit Versions– und Konfigurationsmanagement?
- Was muss man nach einem Change alles Testen?
- Was ist das passende Tool, um die Changes zu managen?

Risikomanagement

Risikomanagement wird oft nicht oder nur halbherzig betrieben. Selbst in sehr großen und risikoreichen Projekten besteht dies oft nur aus einem einmaligen kurzen Brainstorming als „Feigenblatt“ am Anfang des Projekts, damit die Verantwortlichen auch dieses Thema abhaken können.

Dabei ist Risikomanagement, wenn es ernsthaft betrieben wird, ein sehr gutes Instrument für die Erreichung einer effektiven Vorgehensweise in der Entwicklung. Durch die Risikoeinstufung können praktisch sämtliche Aktivitäten im Projekt differenziert betrachtet und vom Aufwand her gesteuert werden. Z.B. kann der Detaillierungsgrad im Requirements-Engineering abhängig vom Risiko gewählt werden, es kann definiert werden, dass bestimmte umfangreiche Testmethoden nur für die Risikoklasse „High“ angewendet werden und Requirements der Risikoklasse „Low“ nur mit Standardfällen getestet werden. Eine gezielte Anwendung kann hier, ohne dass auf eine strukturierte und nachvollziehbare Vorgehensweise verzichtet wird, viel Aufwand und Geld sparen.

Wenn man Risikomanagement durchführen möchte, sollte man sich folgende Fragen stellen:

- Was bringt uns Risikomanagement überhaupt?
- Wozu soll es im konkreten Fall betrieben werden?
- Wie findet man die Risiken im Projekt / in der Entwicklung?
- Wie geht man bei der Risikoerhebung vor?
- Was ist die passende Methode zur Risikoeinstufung?
- Wissenschaftlich oder „Daumen x PI“?
- Welcher Detaillierungsgrad / Ebene soll betrachtet werden?
- Produkte, Teile, Requirements, etc.?
- Was passiert in der weiteren Folge mit der erfolgten Risikoeinstufung?
- Wann wird eine Risikoanalyse im Prozess / Projekt durchgeführt?
- Wird die Risikoanalyse regelmäßig aktualisiert?

Hat das Ergebnis Auswirkungen auf Requirements-Spezifikation, Test-Spezifikation, Test-Automatisierung, Dokumentation, etc. oder wird es einmal durchgeführt, archiviert und dann vergessen?

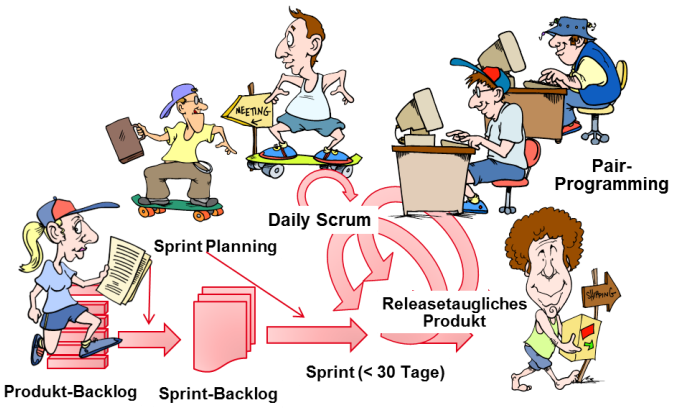
Agile Methoden

Es gibt praktisch kaum eine Software-Entwicklungs-organisation, die sich nicht mit agilen Methoden auseinandersetzt. Abhängig vom Umfeld (Forschung, Neuprojekt, Wartung, kontinuierliche Produktentwicklung, öffentliche Ausschreibung, Festpreise, kreatives Umfeld, sicherheitskritisches Umfeld, etc.) passiert dies in der Praxis jedoch mit mehr oder weniger Erfolg.

Da Agile Methoden meist nur einen Ausschnitt der Realität betrachten (z.B. SCRUM primär Projektmanagement und Projektcontrolling; KANBAN primär das Ressourcenmanagement), ist es notwendig, die agilen Methoden immer im Kontext der gesamten Organisation und der restlichen Prozessthemen zu betrachten.

Nachfolgend sind einige Fragestellungen angeführt, die bei der Einführung und Anwendung agiler Methoden berücksichtigt werden sollten:

- Passt die agile Vorgehensweise für das jeweilige Entwicklungsumfeld? Agile Methoden beinhalten gute Ansätze für viele Bereiche, sind aber kein Allheilmittel.
- Werden die agilen Methoden richtig angewendet und deren Vorteile auch genutzt?

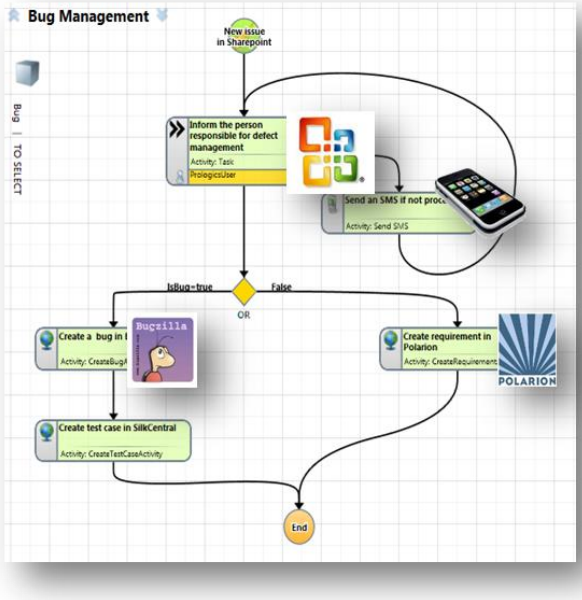


- Werden die agilen Methoden „dogmatisch“ gesehen oder werden Sie pragmatisch im Kontext mit anderen Themen verstanden und angewendet?
- Wie wird das Thema Testautomatisierung in diesem Zusammenhang betrieben?
- Wurde das passende Prozessumfeld geschaffen?
- Wurden die in agilen Methoden nicht abgedeckten Themen wie z.B. Requirements-Engineering, Testen, Change-Management, Configuration-Management, Risiko-Management, Produkt-Management, etc. ebenfalls betrachtet und im agilen Kontext definiert?
- Wie ist die Mentalität der Mitarbeiter, des Managements, der Kunden zu diesem Thema?
- Passt das vertragliche Projekt-Umfeld für diese Methode wie z.B. Budget, zeitliche Vorgaben, Festpreis, Muss-Anforderungen, Ausschreibungen, etc.?
- Werden passende Tools verwendet?

Prozess-Automatisierung und Integration

Es gibt eine Unmenge an verschiedenen Tools im Einsatz in der SW-Entwicklung.

Meist sind dabei viele Medienbrüche und Integrationslücken vorhanden, wodurch die Entwicklungsprozesse oft nicht so effizient laufen, wie dies möglich wäre.



Eine gute Tool-Integration, Prozess-Automatisierung und übergreifende Prozess-Visualisierung ist der Schlüssel zum Erfolg, wobei die heute am Markt vorhandenen Application-Life-Cycle-Lösungen (ALM) insofern meist problematisch sind, als die potentiellen Kunden bei einer Einführung auf breiter Basis sehr oft etablierte Lösungen „wegschmeißen“ müssen.

Außerdem haben ALM-Lösungen oft einen bestimmten Schwerpunkt (z.B. Testmanagement), in dem sie gut sind, die anderen Bereiche sind jedoch oft nicht optimal umgesetzt. Weiters müssen ALM-Lösungen auch noch mit weiteren Systemen wie ERP, Projektmanagement, etc. gekoppelt werden. Leider sind in den ALM-Lösungen und auch in verschiedenen anderen Teil-Produkten die Prozesse oft fest verdrahtet oder nur über Programmierungen anpassbar, was in der Folge zu einem erhöhten Wartungsaufwand und auch Intransparenz in den Prozessen führt.

Ein guter alternativer Ansatz zu ALM ist daher die intelligente Kopplung der benötigten Tools (das für den jeweiligen Themenbereich am besten passende) durch standardisierte Schnittstellen über ein zentrales Steuerungs-System (z.B. können hier etablierte Workflow-Engines dafür verwendet werden).

Dies schafft einen flexiblen und auch wartbaren Prozess, der weniger anfällig ist, wenn in einem Teilbereich eine neue Technologie oder Tool eingeführt wird.

Resümee

- Alle genannten Themenbereiche sind in der Welt der Software Qualität sehr wichtig.
- Die Themen sind komplex und miteinander vernetzt und sollten nicht isoliert betrachtet werden. Das Gesamtoptimum für die Organisation sollte im Fokus stehen.
- Dazu ist es wesentlich, sich nicht nur auf genau ein Thema zu konzentrieren, sondern das Prozess-System
- als Ganzes zu betrachten, zu analysieren und dann passende Maßnahmen zu definieren. Die Umsetzung kann dann natürlich Schritt für Schritt erfolgen.
- Einen Prozess- und Quality-Manager in der Organisation dafür zu etablieren, wird für die nachhaltige Etablierung des Qualitäts-Gedankens hilfreich sein.
- Prozesse und das Umfeld ändern sich laufend, daher ist es sehr wesentlich, ständig dranzubleiben (KVP = Kontinuierliche Prozess-Verbesserung).

Autor

Dipl.-Ing. Johannes Bergsmann

Geschäftsführender Gesellschafter
Berater & Trainer bei Software Quality Lab

Impressum

Der Quality Knowledge Letter ist ein periodisches Informationsmedium von Software Quality Lab und dessen Partnern mit dem Schwerpunkt SW-Qualität.

Weitere Infos zu diesem und anderen Themen finden Sie auf www.software-quality-lab.com

Leistungen von Software Quality Lab

- Analyse und Optimierung der gesamten Entwicklungsprozesskette vom Requirements-Engineering über Architektur und Coding bis hin zum Testen und der Auslieferung
- Analyse und Optimierung der begleitenden Prozesse wie Projektmanagement, Produktmanagement, Risikomanagement, Change- und Configuration-Management, etc.
- Methodische Hilfe bei der Testautomatisierung
- Einführung und Anwendung von Agilen Methoden
- Begleitung von großen Software-Projekten als Prozess- und Qualitätsmanager
- Seminare für Entwickler, Tester, Architekten, Entwickler, Projektmanager, Produktmanager und Anwender
- Operative Unterstützung in Projekten und Entwicklungsvorhaben