Compliance in agilen Projekten umsetzen

Alles geregelt

Tilo Linz

Für Softwarehäuser gelten wie für andere Unternehmen Compliance-Anforderungen und Normen. Beispielsweise muss sich jedes Softwareartefakt lückenlos rückverfolgen lassen. Ebenfalls muss klar sein, welche Anforderungen es umsetzt und wie getestet wurde. Das ist auch beim agilen Entwickeln möglich, jedoch nicht einfach.

Fast alle Softwarehäuser unterliegen externen Nachweis- und Compliance-Anforderungen. So erwarten größere Kunden in aller Regel die Anwendung und den Nachweis eines nach ISO 9001 zertifizierten Qualitätsmanagementsystems. Hersteller von Produkten und elektronischen Systemen, in denen Software sicherheitsrelevante Aufgaben übernimmt, müssen IEC 61508-3 beachten. In Branchen wie der Automobilindustrie, der Medizintechnik, der Bahntechnik oder der Luftfahrt sind darüber hinaus zahlreiche branchenspezifische Normen einzuhalten. Können Unternehmen, die von derartigen Normen und Standards betroffen sind, die sich ergebenden Compliance-Anforderungen erfüllen, wenn agil entwickelt wird?

Keine der genannten Normen erzwingt oder fordert einen klassischen Entwicklungsprozess, etwa nach dem V-Modell. Durchgehend geben sie aber vor, dass die Softwareentwicklung als spezieller, qualitätsrelevanter Engineering-Prozess nach einem definierten Vorgehen abläuft. Das bedeutet: Der Prozess muss schriftlich dokumentiert sein und für jedes Entwicklungsprojekt muss nachvollziehbar und jederzeit belegbar sein, dass er in der Praxis tatsächlich befolgt wird. Durch die Aufnahme der agilen Vorgehensweise als neue oder zusätzlich zulässige Vorgehensweise im QM-System lässt sich diese Forderung einfach erfüllen (siehe Kasten „Kultur des Qualitätsmanagements verändern“). Der Artikel gibt einige Hinweise, was man dabei beachten sollte. Darüber hinaus können, je nachdem welche Bereiche des Unternehmens ebenfalls agile Methoden adaptieren, weitere Unternehmensprozesse betroffen sein.

Einige Normen formulieren spezifische Anforderungen an einzelne Entwicklungsaktivitäten beziehungsweise Prozessschritte. Beispiele sind die Definition von Risikoklassen (etwa in EN 62304) oder Sicherheitsanforderungsstufen (etwa in ISO 26262) verbunden mit Anforderungen an Softwaredesign und Entwicklungsmethoden, darunter Softwaretests. Der Entwicklungsprozess muss entsprechende Schritte enthalten, die die jeweiligen Forderungen zuverlässig umsetzen. Es muss also eine Risiko- und/oder Sicherheitsklassifikation stattfinden, und in Folge muss der Prozess sicherstellen, dass jedes Softwareartefakt gemäß seiner spezifischen Risiko-/Sicherheitseinstufung entwickelt und geprüft wird.

Risiko- und Sicherheitsklassifikation

Spezifische Anforderungen an einzelne Entwicklungsaktivitäten lassen sich im Grunde unkompliziert abbilden, nämlich über eine geeignete „Definition of Done“ beziehungsweise durch entsprechend formulierte „Fertig“-Kriterien für die betroffene Entwicklungsaktivität. Ist beispielsweise eine Funktion mit hoher Sicherheitsstufe zu implementieren, so wird die geforderte Testabdeckung als Done-Kriterium der Implementierungsaufgabe mitnotiert.

So weit, so einfach. Schwieriger ist es, über den Prozess sicherzustellen, dass in jedem Sprint jede Implementierungsaufgabe im Rahmen einer Impact-Analyse bezüglich ihrer Sicherheitsstufe bewertet wird und immer ein passendes Done-Kriterium erhält. Hier kann eine zuverlässig angewendete Checkliste helfen oder vorgedruckte Taskkarten, auf denen „generische“ Done-Kriterien (wie Coverage-Limits) vorgedruckt und ankreuzbar stehen. Eventuell ist es besser, kritische Prozessschritte, die nicht ausgelassen werden dürfen (wie „Impact-Analyse durchführen“), am Taskboard durch eine entsprechende Spalte darzustellen. Wie man dies prozessseitig gelöst hat, ist auf jeden Fall ein Punkt, der in der Prozessbeschreibung des QM-Systems stehen muss und von dem einzelne Teams nicht abweichen dürfen.

Überall dort, wo sicherheitskritische Produkte entstehen, fordern die einschlägigen Normen, dass Designentscheidungen und Produktänderungen über den Entwicklungs- und Lebenszyklus hinweg nachvollziehbar und rückverfolgbar sind. Die Philosophie dahinter: Das Produkt muss sicher funktionieren. Die ausreichende Sicherheit muss durch entsprechend gestaltete Produkteigenschaften beziehungsweise Funktionen im Rahmen des Produktdesigns festgelegt und im Entwicklungsprozess realisiert werden. Durch geeignete Tests ist nachzuweisen, dass das Produkt und insbesondere seine sicherheitsrelevanten Eigenschaften/Funktionen wie geplant funktionieren.

In der Praxis bedeutet dies, dass für jedes Softwareartefakt lückenlos rückverfolgbar sein muss, welche Produktanforderungen es umsetzt, wie dies gemäß seiner Spezifikationen erfolgt und durch welche Testfälle mit welchem Testergebnis die korrekte Umsetzung und Funktion nachgewiesen wurde. Änderungen an Design oder der Implementierung des Produkts dürfen seine Sicherheit ebenfalls nicht unbeabsichtigt reduzieren oder aufheben. Das impliziert, dass vor jeder Änderung eine Analyse (impact analysis) stattfinden muss, die herausfinden soll, welche Auswirkungen auf die Produktsicherheit die Änderung hat beziehungsweise hätte und welche Maßnahmen gegebenenfalls nötig sind, um die Sicherheit aufrechtzuerhalten. Jede Änderung muss, ausgehend von ihrem Anlass über das Ergebnis der Impact-Analyse bis zum Ergebnis der Modifikation, gleichfalls rückverfolgbar sein.

Lückenlose Rückverfolgbarkeit notwendig

Es ist nicht ganz einfach, diese Traceability in einem agilen Entwicklungsprozess zu gewährleisten. Denn naturgemäß kommen Änderungen hier häufig vor. Wenn der Traceability-Mechanismus umständlich ist oder unzuverlässig arbeitet, erzeugt das schnell großen bürokratischen Buchführungsaufwand oder die Rückverfolgbarkeit bleibt lückenhaft. Des Weiteren muss sich das Team Gedanken machen, auf welcher Granularitätsstufe Rückverfolgbarkeit nachgewiesen werden muss. Als Minimum ist eine Verbindung zwischen Produktanforderungen und Systemtestfällen herzustellen, sodass sich anhand der erfolgreich gelaufenen Tests zeigen lässt, dass jede Anforderung erfüllt wird. Dazu gehört, dass begründbar ist, warum einer oder mehrere Systemtestfälle die Erfüllung eines bestimmten Requirements angemessen validieren. Im Prozess muss daher ein Review erfolgen, das die Angemessenheit der Systemtestfälle überprüft. Wenn Testfälle Fehler aufdecken, muss ein zuverlässiger Bewertungs- und Korrekturprozess einsetzen. Er läuft in der Regel über ein Fehlermanagementverfahren. Um eine Fehlermeldung zu ihrer Ursache zurückverfolgen zu können, muss die Meldung wiederum mit dem Testfall verknüpft sein. In der Praxis gelingt das Sicherstellen von Rückverfolgbarkeit nur durch Einsatz geeigneter miteinander gekoppelter Werkzeuge für Anforderungs- sowie Testmanagement.

Agiles Team in der Zwickmühle

Eine auf den ersten Blick gute Nachricht ist, dass dieser Formalismus prinzipiell nur für auszuliefernde Produktstände erforderlich ist. Einem agilen Team hilft das jedoch nicht viel. Denn ob es ein Ergebnis ausliefert, entscheidet sich oft erst am Ende des Sprints. Die Tracability-Informationen nachzudokumentieren ist nicht zu empfehlen. Außerdem müsste man sicherstellen, dass Produktstände, die ohne Tracability-Mechanismus entstanden sind, keinesfalls versehentlich in die Auslieferung gelangen. Sinnvoller ist, wenn der Mechanismus als Routinehandlung verankert und in jedem Sprint praktiziert wird.

Ob Tracability auf der Ebene „Requirements-Systemtests“ ausreicht oder ob feingranularere Softwareartefakte miteinbezogen werden müssen, hängt davon ab, auf welcher Dokumentationsebene Designentscheidungen fallen und auf welcher Ebene der Softwarearchitektur die entsprechenden Funktionen liegen und damit die zugehörigen Testfälle angesiedelt sind. Die Traceability-Kette kann also durchaus – zumindest punktuell – bis zu den Unit Tests hinabreichen.

Von Compliance-Vorschriften betroffen sind nicht nur Softwareunternehmen. Anwendungen in der Unternehmens-IT können ebenfalls Compliance-Anforderungen unterliegen, zum Beispiel an Informationssicherheit und Datenschutz, an die Archivierung von Daten, inklusive Fristen zur Datenaufbewahrung

Kultur des Qualitätsmanagements verändern

Klassisches Qualitätsmanagement funktioniert top-down. Ein zentraler QM-Stab ist dafür verantwortlich, die Vorgehensweisen zu dokumentieren und – soweit sinnvoll – zu standardisieren. Er schreibt die Prozesshandbücher und liefert die standardisierten Prozesse an alle betroffenen Unternehmensbereiche aus. Agile Teams funktionieren bottom-up. Das Team wählt und bestimmt seine interne Vorgehensweise selbst.

Dem muss sich der QM-Stab anpassen: Aus einer Stelle, die Prozesse vereinheitlicht, wird eine Gruppe, die den Teams hilft, die selbst gewählten Abläufe zu dokumentieren. Aus einer zentralen Standardisierungsinstitution entsteht eine Gruppe, die den Erfahrungsaustausch unter den Teams fördert und Methoden publiziert, die sich bewährt haben. Umgekehrt schlägt sie für Dinge, die sich nicht bewähren, Alternativen vor.

Der QM-Stab entwickelt sich zum Ratgeber und zur Serviceeinheit. Er betrachtet die agilen Teams als Kunden, die Beratungsleistung anfordern können. Will ein Team zum Beispiel seinen Continuous-Integration-Prozess verbessern und hat erkannt, dass dazu ein besseres CI-Tool notwendig ist, kann es den QM-Stab beauftragen, ein Werkzeug auszuwählen, vorzustellen und einzuführen. Das Team konzentriert sich währenddessen auf seine Entwicklungsarbeit.

Interessen der Teams ausbalancieren

Wenn es mehrere agile Entwicklungsteams gibt, lässt sich nicht mehr so einfach delegieren. Der QM-Stab muss in diesem Fall die Interessen der verschiedenen Teams ausbalancieren – untereinander und mit den Interessen der Gesamtorganisation. Am Ende entscheiden aber auch hier die Teams, ob und welche Empfehlungen sie annehmen, selbst wenn sie dadurch eine angedachte Standardisierung verhindern. Diesem Realitätscheck beziehungsweise Akzeptanztest muss sich der QM-Stab stellen. Wenn das Team eine Empfehlung ablehnt oder nicht nutzt, ist es Aufgabe der QM-Leute, zu überlegen, woran das liegt und was sie gegebenenfalls besser erledigen können.

Der QM-Stab sollte die Gelegenheit nutzen und seine eigene Arbeitsweise agil organisieren, beispielsweise nach einem an Kanban orientierten Vorgehen. Das verschafft ihm eine bessere Reaktionsfähigkeit, mehr Schnelligkeit und damit die Lieferfähigkeit, die nötig ist, um von den agilen Projektteams weiterhin (oder wieder) akzeptiert zu werden. Lieferfähigkeit bedeutet hier: In der Lage sein, ein „shippable product“ abzuliefern, zum Beispiel eine Prozessbeschreibung, die die Projektteams zu hundert Prozent akzeptieren und als nützlichen Beitrag schätzen. Oder die schnelle Einführung eines Tools inklusive aller nötigen Validierungsschritte ohne Behinderung der nutzenden Teams.

Dieser Aufgabenkatalog des QM-Stabs überschneidet sich teilweise mit dem eines Scrum Master. Beide verfolgen ähnliche Ziele: Der QM-Stab stellt sicher, dass das ganze Unternehmen die Spielregeln des QM-Systems richtig umsetzt. Der Scrum Master ist dafür verantwortlich, dass „sein“ Team die Scrum-Praktiken richtig anwendet. Der Unterschied liegt im Fokus (Unternehmen versus Team) und im Instrumentarium (ISO 9000 versus Scrum).

QM-Stab als Dienstleister für Teams

Im Zuge der skizzierten Veränderung des QM-Systems und der QM-Kultur zu einem agilen QM-System können diese Unterschiede abnehmen oder ganz verschwinden. Der QM-Stab agiert als Dienstleister für jedes Team, der Scrum Master vertritt nicht nur sein Team, sondern hat den Gesamterfolg im Blick. Und beide nutzen das agile QM-System als gemeinsames Instrumentarium. Ergebnis dieses Transitionsprozesses kann es sein, dass die Mitarbeiter des QM-Stabs und die Scrum Master (in der Linienorganisation) in einem gemeinsamen, interdisziplinären agilen QM-Team zusammengefasst werden. Das neue Team besitzt alle Kompetenzen, um Scrum-Teams methodisch wirksam zu unterstützen: Expertise über die für das Unternehmen relevanten Normen, Know-how und Tools zur Prozessmodellierung, ausgebildete Assessoren und Auditoren, Spezialisten und Trainer für die relevanten Techniken und natürlich zertifizierte Scrum Master.

sowie Mechanismen zu deren Wiederherstellung. In Branchen, die sicherheitskritische Software herstellen, müssen Werkzeuge für den Engineering-Prozess vor erstmaliger Inbetriebnahme und unter Umständen nach Updates anhand bestimmter Vorschriften validiert werden.

Für die Hersteller der Software bedeutet das, dass sie die für ihre jeweiligen Kunden geltenden Vorschriften kennen müssen und sie durch entsprechende Produktfeatures oder passende Support- oder Serviceangebote unterstützen. Produktfeatures, die nötig sind, um bestimmte Normforderungen umzusetzen, müssen prozessseitig nicht anders behandelt werden als „normale“ Features. Vermutlich dürften sie aber eine tendenziell höhere Priorität erhalten. Ergänzende Support- oder Serviceangebote können im agilen Entwicklungsprozess durch die enge Zusammenarbeit mit dem Kunden und die frühe Verfügbarkeit einsatzfähiger Produktversionen eventuell frühzeitiger und besser identifiziert werden als bei einer klassischen Entwicklung.

Manche Software muss validiert werden

In manchen Branchen fordern Regularien (etwa FDA OTS, FDA Validation in der Medizintechnik), dass Standardsoftware, die ein Hersteller in seine Produkte integriert, aber auch Programme, die als Werkzeuge im Engineering-Prozess oder in der Produktion zum Einsatz kommen, vor der Übernahme in den produktiven Einsatz geprüft werden, um sicherzustellen, dass die von Dritten zugelieferte Software im konkreten Einsatzumfeld den beabsichtigten Zweck erfüllt. Wird die betroffene Software agil entwickelt, kann dies das Entwicklungsteam empfindlich bremsen.

Denn eine solche Validierung erzeugt beim Kunden erhebliche Arbeit. Unter Umständen muss er jedes neue Produktrelease oder gar jeden Patch validieren. Um den Aufwand zu minimieren, ist der Auftraggeber an langen Releasezyklen interessiert. Liefert das agile Team zu häufig, nimmt er die Software vielleicht nicht in Betrieb oder überspringt Releases. In beiden Fällen erhält das Team über die ausgelassenen Releases kein brauchbares Feedback. Möglicherweise fühlt er sich sogar unangenehm unter „Update-Druck“ gesetzt. In solchen Konstellationen ist ein klassisches V-Modell unter Umständen die für den Kunden günstigere Vorgehensweise. (jd)

Quelle

Der Artikel ist ein überarbeiteter und gekürzter Auszug aus diesem Buch:

Tilo Linz

Testen in Scrum-Projekten, Leitfaden für Softwarequalität in der agilen Welt

dpunkt.Verlag 2013