- Zwischenergebniseliminierung

Gesamtabschluss

- Gesamtergebnisrechnung
- Gesamtbilanz
- Gesamtfinanzrechnung (Kapitalflussrechnung)
- Anlagen zum Gesamtabschluss

Momentan werden noch Software-Anwender aus diesen Bereichen gesucht, die bereit sind, ihre in den vergangenen Jahren gesammelten Erfahrungen beim Umgang mit entsprechenden Programmen in die Kriterien von GA.B einfließen zu lassen.

Ansprechpartner ist Herr Wolf, erreichbar unter wolf@okksa.de.

Fachprogramme auf den Prüfstand Teil 15: Softwareprüfung und -freigabe im Kontext Agiler Softwareentwicklung

(von Dr. Uwe Schwochert)

Zusammenfassung

Teil 14 der Artikelserie "Fachprogramme auf dem Prüfstand" stand unter dem Titel "Was muss ein Programm über sich verraten?" In Anbetracht der immer öfter an den Autor herangetragenen Fragstellungen zu agiler Softwareentwicklung wird hier näher auf die Problematik von Prüfung und Freigabe in diesem Umfeld eingegangen. Es werden Lösungsansätze und deren Auswirkungen diskutiert.

Was ist agile Softwareentwicklung?

Mit agilen Methoden wie "Scrum" werden in der Softwareentwicklung neue Prioritäten gesetzt. Statt dem sturen Befolgen eines vorab festgelegten Plans oder Pflichtenheftes werden Interaktion zwischen Entwickler und Anwender sowie Flexibilität in den Vordergrund gestellt. Damit soll erreicht werden, dass seitens der Softwareentwicklung besser auf komplexe und sich öfter ändernde Anforderungen der Anwender eingegangen werden kann.

Grundfundament der agilen Softwareentwicklung sind die Werte des "Agilen Manifests" von 2001. Nach diesen stehen

- Individuen und Interaktionen über Prozessen und Werkzeugen,
- funktionierende Software über einer umfassenden Dokumentation,
- Zusammenarbeit mit dem Kunden über der Vertragsverhandlung,
- Reagieren auf Veränderung über dem Befolgen eines Plans.

Diese Werte drücken sich aus

1. im "agilen Prinzip",
2. der "agilen Methode" und
3. dem "agilen Prozess"[1].

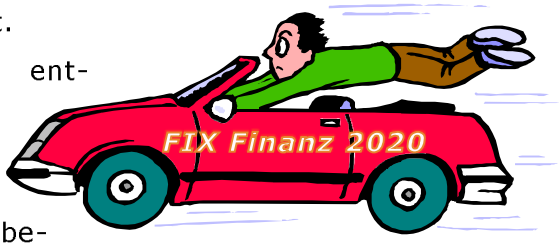
Ein wichtiges Ziel ist, neben der Flexibilität, auch die Entbürokratisierung der Softwareentwicklung, die flache Aufwandskurven ermöglichen soll.

In Anbetracht der Marktherausforderungen stellen immer mehr große Softwarehäuser Kernbereiche ihrer Entwicklungsabteilungen auf agile Prozesse um. Und damit erreichen agil entwickelte Programme auch konservative Nutzerkreise, deren Anforderungslandschaft ansonsten nicht durch große Anforderungsdynamik geprägt ist.

Die zunehmende Online-Orientierung auch im Bereich komplexer Fachanwendungen fördert diesen Effekt und ermöglicht neue Ansätze wie "continuous delivery" und "continuous deployment"[2].

Was bedeutet agile Softwareentwicklung für den Fachanwender?

Viele Fachanwender, z. B. in den Finanzabteilungen der Kommunen, haben wahrscheinlich schon einmal Erfahrungen mit dynamischen Softwareentwicklungsprozessen gemacht. Deutlich spürbar ist der Effekt, wenn man z. B. durch eine Fehlermeldung, eine neue Anforderung oder auch direkt als "Pilotanwender" eines neuen Programms plötzlich in unmittelbarem Kontakt zur Entwicklungsabteilung oder zu einzelnen Softwareentwicklern steht. Hier hat dann ein "einfacher" Anwender direkten und schnellen Einfluss auf die Gestaltung eines Programms oder Features. Gleichzeitig ergibt sich in so einem "Pilotbetrieb" eine mehr oder weniger starke Unsicherheit bzgl. der (je nach Anwendungssituation) mehr oder weniger verbindlichen Softwarefunktionalität.



Genau hier entsteht dann auch das Problem, wenn dieser "Pilotbe-

[1] Details s. https://de.wikipedia.org/wiki/Agile_Softwareentwicklung.

[2] Definitionen s. https://www.scrum.de/continuous-delivery-definitionen/

trieb" zum Regelfall wird. Wenn also hochdynamische Entwicklungsprozesse Grundlage der bereitgestellten Softwarefunktionalität werden.

In funktional überschaubaren Softwarewelten könnte das Prinzip "Ruf an, wenn dir etwas nicht passt, wir reagieren schnell" durchaus als Vorteil gesehen werden. Im Bereich der rechtsverbindlichen oder sensiblen Datenverarbeitung, ja eigentlich überall dort, wo Software Regeln mit hoher Komplexität und Tragweite umsetzen muss, fehlt dem Anwender die Möglichkeit, Fehler sofort zu erkennen oder korrigieren zu können.

Hier ist weniger eine gute Kommunikation mit dem Entwickler oder Flexibilität gefragt sondern mehr dessen Kenntnis all der Regeln, die die Programme beachten müssen. Die Kontrolle dieser Regelkonformität ist anwenderseitig kein laufender Prozess, sondern eher ein "sich auf etwas verlassen können"; eine Kontrolle der richtigen Funktionalität kann häufig nur punktuell stattfinden. Die "User Story" (im Sinne von Scrum) müsste in etwa lauten: "Mein Programm soll Vorgänge nach einer Vielzahl von Regeln meines Tätigkeitsgebietes verarbeiten, die ich im Zweifelsfall selbst nicht alle kenne."

Fachanwender sind aus diesem Grund nicht unbedingt Anhänger der agilen Prinzipien. In Anbetracht der komplexen Funktionalität ihrer Programme vertrauen Sie eher auf detaillierte Pflichtenhefte/Anforderungskataloge und entsprechende vertragliche Zusagen als auf schnelle Releasewechsel.

Problematik der Prüfung und Freigabe im Kontext agiler Softwareentwicklung

Es wird schnell klar, dass dem Fachanwender eine einmalige oder gelegentliche Prüfung der Umsetzung seiner komplexen Funktionsansprüche nicht reicht, um die Funktionsrisiken, die durch die dynamische Weiterentwicklung seiner Software entstehen, in den Griff zu bekommen. Eine gestern festgestellte Korrektheit des Ausdrucks eines Jahresabschlusses kann schon morgen durch ein "agil" eingespieltes Update, welches eigentlich nur das Seitenformat betreffen sollte, in Mitleidenschaft gezogen werden.

Eine "Vollprüfung" der gesamten Anwendung ist in einem so kurzen Zyklus ebenfalls nicht machbar, da dies in Anbetracht der Gesamtkomplexität der Programmfunktionen einfach nicht praktikabel ist: Vor dem Abschluss der "Vollprüfung" würden bereits neue Updates und Patches des "dynamischen" Entwicklerteams deren Ergebnis obsolet machen.

Es ergibt sich ein Widerspruch zwischen klaren (und oft regulatorisch vorgegebenen) Funktionsanforderungen der Anwender und der Möglichkeit des Nachweises ihrer Umsetzung. Das Problem ist dabei nicht primär mit dem agilen Entwicklungsprozess verbunden, sondern entsteht durch die mit diesem Prozess verbundene Dynamik.

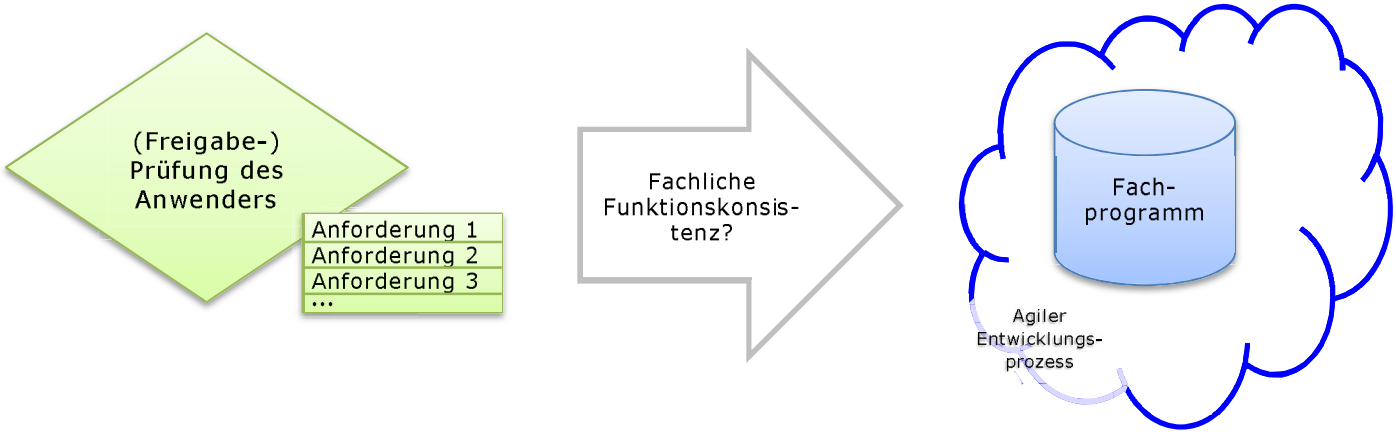
Lösungsansätze

Es liegt auf der Hand, dass sich aus der Verbreitung agiler Entwicklungsprinzipien Bedarf für neue Prüfansätze ergibt.

Ansatz 1

Eine Variante könnte sein, dass zwischen Entwickler und Anwender eine Zwischeninstanz installiert wird, die die Funktion hat, das in Frage gestellte Vertrauen in die Programmfunktionalität aufrecht zu erhalten. In Kenntnis der detaillierten Nutzeranforderungen und mit direktem Zugriff auf die Software-Entwicklungsprozesse müsste dieser Vertrauensprovider dafür sorgen, dass z. B. beim Verbessern von Softwarefunktion A die Softwarefunktion B nicht in Mitleidenschaft gezogen wird. Dieser Vertrauensprovider wäre quasi ein organisatorisch eingefügter "Überwacher" des Anforderungsmanagements beim Entwickler.

Der notwendige qualifizierte Zugriff auf den Entwicklungsprozess dürfte allerdings nur in



unmittelbarer "Nähe" zu diesem gewährleistet sein, der Vertrauensprovider müsste Teil des Softwareentwicklungsunternehmens oder zumindest einflussreicher langfristiger Partner sein. Damit wird es mittelfristig schwer fallen, die für das Anwendervertrauen benötigte Unabhängigkeit aufrecht zu halten.

Ansatz 2

Die in Ansatz 1 notwendige Rolle des Vertrauensproviders läuft darauf hinaus, dass dieser unter genauer Kenntnis der Entwicklungsprozesse beurteilen kann, welche Weiterentwicklungen am Programm ("sprints") Einfluss auf welche Funktionalitäten des Gesamtprogramms haben könnten. Dabei geht es also um die Überwachung der Nutzeranforderungen ("product backlog") und ihrer Abhängigkeiten ("requirements traceability").

Diesen Zusammenhang müsste genau genommen auch der Entwicklungsprozess selbst liefern, wenn ein entsprechendes Management vorausgesetzt wird. In der Integration des "Vertrauensmanagements" in den Entwicklungsprozess liegt also eine weitere, vielleicht wirksamere Herangehensweise an das Prüfungsproblem.

Wenn nämlich die verschiedenen Einzelfunktionen eines Programms eine so zentrale Rolle aus Sicht der Prüfbarkeit und letztlich der Nutzbarkeit des Programms spielen, so muss dies auch Maßgabe für den wie auch immer gearteten Entwicklungsprozess sein.

Es ergibt sich die Besonderheit (aus Sicht der agilen Entwicklung), dass die Einhaltung des dem Anwendungsfeld zu Grunde liegenden Regelsystems und der Nachweis darüber primärer Zweck der Entwicklung ist.

Ein temporär nicht funktionierender Jahresabschluss wäre also im Zweifelsfall eher verkraftbar, als einer, der die Berechnungsregeln falsch anwendet oder auch nur eine diesbezügliche Unsicherheit aufwirft.

Es bedarf hier eines Entwicklungsmanagement-Systems, welches neben den Vorteilen agiler Arbeitsweisen auch die unbedingte funktionale Sicherheit im Auge hat. Dadurch steht jedes Modul, jede Änderung daran ("Sprint") im Kontext des jeweils berührten Funktionsspektrums. Die Verfolgbarkeit der funktionalen Abhängigkeiten wird ein zentrales Element der Entwicklung.

© Fotomek, Fotolia.com

Sicherstellung der Funktionskonsistenz (und in der Folge eine funktionalen Resilienz) meint dabei nicht nur, dass z. B. alle potentiellen Berechnungen des (Finanz-)Programms bei jedem Entwicklungsschritt entwicklerseitig kontrolliert werden. Genauso wichtig sind die "Softskills" des Programms, wie korrekte Darstellung von Werten auf Bildschirm und Druck, Benutzerverwaltung usw. (s. OKKSA FÜ.B). Nicht alle Funktionen können automatisiert getestet werden; insbesondere die Überwachung der Softskills erfordert auch weitere Testansätze.

Das "Einbauen" eines allgegenwärtigen Funktionsrasters in die Elemente der Softwareentwicklung ist nur ein Schritt aus Sicht der betroffenen (funktionssensiblen) Anwender. Dort wo sie verantwortlich für die Nachweise zur Funktionssicherheit der Software sind (und diese Verantwortung ergibt sich z. B. aus den meisten Haushaltsverordnungen), brauchen Sie auch die Möglichkeit, die Funktionskonsistenz der angewendeten Software zu überwachen. Dies kann ihnen in dieser Dynamik jedoch nur gelingen, wenn der Entwicklungsprozess die entsprechenden Maßnahmen seines Anforderungsmanagements auch dokumentiert und somit prüfbar macht.

Praktische Umsetzung aus Sicht des Anwenders

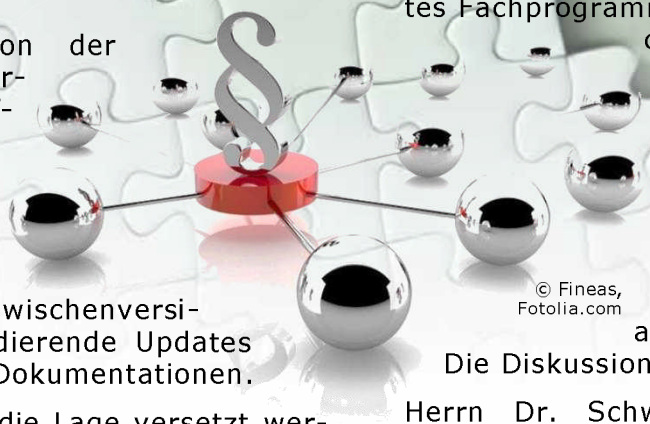
Wichtig ist, zunächst noch einmal den besonderen Bedarf der Fachanwender im Blick zu haben:

1. breite regel- bzw. gesetzesbasierte Funktionalität,
2. verlässliches Befolgen der Regeln,
3. Nachprüfbarkeit.

Elemente der funktionalen Qualitätssicherung müssen damit Teil des "Outputs" werden, sichtbar und überprüfbar für den Anwender. Konkret könnte dies bedeuten:

1. Transparenz hinsichtlich des der Entwicklung und ihrer Qualitätssicherung zu Grunde liegenden Funktionsrasters gegenüber dem Anwender.
2. Dokumentation der Qualitätssicherungsmaßnahmen, die die Funktionssicherheit gewährleisten sollen.

3. Klares, für den Anwender begreifbares Versionsmanagement (Versionsklarheit, Versionsnummernsystem, Versionsdokumentation).
4. Klare Dokumentation der mit einem Patch/ Version jeweils betroffenen Funktionalitäten.
5. Schaffung der Möglichkeit des Überspringens von (nicht benötigten) Zwischenversionen durch konsolidierende Updates und entsprechende Dokumentationen.



© Fineas, Fotolia.com

Der Anwender sollte in die Lage versetzt werden, die jeweils direkt oder indirekt betroffenen Funktionalitäten zu überblicken und nach eigener Maßgabe bei Bedarf zusätzliche Überprüfungen anzustoßen. Noch wichtiger dürfte jedoch der resultierende Vertrauensgewinn in den Entwicklungsprozess und die ihm innewohnenden Funktionssicherungsmechanismen sein.

Auswirkungen des Vorgehens auf Prüfung und Freigabe

Nach wie vor ist eine regelmäßige Grundprüfung der Programme (mit Blick auf die seit der letzten Prüfung vorgenommenen Änderungen) notwendig und sinnvoll. Eine auf jedes einzelne Update bezogene Prüfung wird jedoch nicht möglich sein. Die punktuelle Prüfung muss deshalb die Funktionssicherung der Softwareentwicklung mit im Auge haben.

Was bedeutet das für die durch OKKSA unterstützte Prüfung und Freigabe, für OKKSA Kriterienkataloge und die entsprechenden Prüfverfahren?

Ganz allgemein wird zunächst klar, dass das Update-Verfahren, die Häufigkeit von Updates, die Deklaration von vorgenommenen Programmänderungen (und den betroffenen Programmbereichen) eine größere Rolle spielen wird. Mögliche Konsequenzen für OKKSA basierte Prüfungen könnten sein:

1. Integrationen von Anforderungen an die Dokumentation des entwicklerseitigen Qualitätsmanagements in die Anforderungskataloge.
2. Änderung des Zyklus von Rezertifizierungen der Programme.
3. Schaffung der Möglichkeit von schlanken "Teilprüfungen" hinsichtlich der jeweils betroffenen Funktionsbereiche.

Änderungen könnten also sowohl die OKKSA Prüfkriterien (insbesondere FÜ.B) als auch das darauf basierende TÜV-Prüfverfahren "Geprüftes Fachprogramm" betreffen. Ob und mit welchen der genannten Maßnahmen dem Problem der Funktionsunsicherheit auf Anwenderseite am besten begegnet werden kann, wird noch zu ermitteln sein. Dies ist eine der Fragstellungen für die entsprechenden OKKSA Gremien, letztlich auch für alle OKKSA Center. Die Diskussion ist eröffnet.

Herrn Dr. Schwochert erreichen Sie über schwochert@trustbit.de.

Geänderte Satzung des OKKSA e. V.

In der Mitgliederversammlung des OKKSA e. V. am 22.06.2016 wurde die Satzung des Vereins geändert bzw. neu verabschiedet. Wir möchten Ihnen die neue Satzung nachfolgend im Wortlaut vorstellen.

Vereinssatzung des OKKSA e. V.

(In der Fassung des Beschlusses der Mitgliederversammlung vom 22.06.2016)

§ 1 Name und Sitz

(1) Der Verein führt den Namen "OKKSA e. V." OKKSA steht dabei als Kürzel für „Offener Katalog kommunaler Softwareanforderungen".

(2) Der Verein hat seinen Sitz in Dresden.

§ 2 Zweck

(1) Zweck des Vereins ist

(a) die Abstimmung und Harmonisierung von Vorgaben an fachbezogene IT-Lösungen zwischen verschiedenen Anwendern und Entwicklern,

(b) die Verbesserung der Softwarequalität und –konformität im fachspezifischen Einsatz, speziell in der Öffentlichen Verwaltung,

(c) die Förderung des Vertrauens in Softwareprodukte.

(2) Der Satzungszweck wird verwirklicht insbesondere durch

(a) die Bereitstellung einer Plattform zur Diskussion aller aus den genannten Vereinszwecken erwachsenden Problemfelder,