SEQIS QualityNews

Analyse, Test und Management

Ausgabe H1/2019

Toolchain

Problem oder Teil der Lösung?

Titel: „Der See im Wald", Künstlerin: Dagmar Andel, Technik: Aquarell und Tusche

Analysis. Test. Management. Better IT Results.

Lesen Sie in dieser Ausgabe:

Ihre Meinung ist gefragt!

Nach den QualityNews ist bekanntlich vor den QualityNews! Schon bald arbeiten wir wieder auf Hochtouren an der nächsten, spannenden Ausgabe. Lesen Sie nur das, was Sie wirklich interessiert! Sagen Sie uns, welche Themen Sie spannend finden.

Kontaktieren Sie uns: marketing@SEQIS.com
Join us: twitter.com/SwTestIsCool

Wir freuen uns auf Ihre Vorschläge und Wünsche!

Über SEQIS QualityNews:
Dieses Magazin richtet sich an Gleichgesinnte aus den Bereichen Software Test, IT Analyse und Projektmanagement im IT Umfeld. Die SEQIS Experten berichten über ihre Erfahrungen zu aktuellen Themen in der Branche. Die Leser des Magazins gestalten die Ausgaben mit: Schreiben Sie uns Ihre Meinung im SEQIS Blog (www.SEQIS.com/de/blog-index) oder als Leserbrief.
Wenn Sie dieses Magazin abbestellen möchten senden Sie bitte ein Mail an marketing@SEQIS.com.

Impressum:
Information und Offenlegung gem. §5 E-Commerce-Gesetz und §25 Mediengesetz

Herausgeber: SEQIS GmbH,
Neusiedler Straße 36, A-2340 Mödling
Tel: +43 2236 320 320 0
Fax: +43 2236 320 320 350
info@SEQIS.com, www.SEQIS.com
Gericht: Bezirksgericht Mödling
Firmenbuchnummer: 204918a
Umsatzsteuer-ID: ATU51140607
Geschäftsführung: Mag. (FH) Alexander Vukovic, Mag. (FH) Alexander Weichselberger, DI Reinhard Salomon

Druck: druck.at Druck- und Handels-gesellschaft mbH, 2544 Leobersdorf
Erscheinungsweise: 2x pro Jahr


Mag. (FH) Alexander Vukovic


Mag. (FH) Alexander Weichselberger


DI Reinhard Salomon

Editorial

Sehr geehrte Leserin,
sehr geehrter Leser,

wir freuen uns, Ihnen die erste Ausgabe der SEQIS QualityNews im Jahr 2019 zu präsentieren.

Wir bedanken uns für die positiven Rückmeldungen auf unsere QualityNews Ausgaben aus 2018 zu den Themen AgilePM und Blockchain. Wir hoffen, Sie konnten interessante Informationen daraus mitnehmen und auch das eine oder andere Mal schmunzeln. Über weitere Anregungen, Themenwünsche und Feedback Ihrerseits freuen wir uns.

In dieser Ausgabe stellen wir unseren technischen Beiträgen wieder einen nicht-technischen Aspekt an die Seite:

Im Heft finden Sie einige Kunstwerke der Lebenshilfe Niederösterreich der Werkstätten Baden und Mödling. Denn nicht nur unsere Spezialisten, sondern auch die Klienten der Lebenshilfe leben für ihre(n) Beruf(ung).

In dieser Ausgabe dreht sich alles um das Thema Toolchain. Per Definition ist die Toolchain „in der Software Entwicklung eine systematische Sammlung von Werkzeug-Programmen, welche zur Erzeugung eines Produktes Verwendung findet. Die Bezeichnung erklärt sich damit, dass die Werkzeug-Programme in der Regel in Form einer Kette nach-einander eingesetzt werden."

Auf den folgenden Seiten geben Ihnen unsere Experten einen Einblick in die vielseitigen Aspekte der Toolchain, bspw. in der IT-Analyse, für Software Tester oder DevOps. Sie zeigen dabei die enormen Vorteile, die bei effizientem Einsatz der richtigen Werkzeuge in der Praxis gewonnen werden konnten, auf.

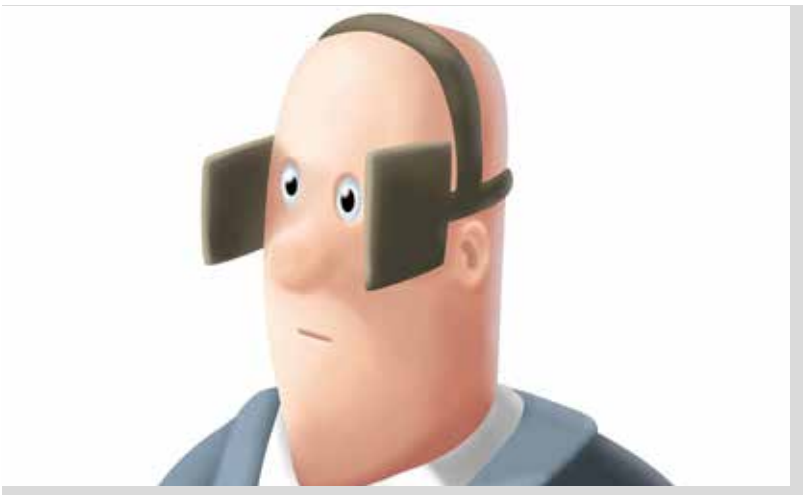
Wir wünschen viel Lesevergnügen mit unseren SEQIS QualityNews!

Ihre SEQIS Geschäftsleitung

Neulich im Netz: Scheuklappen für Arbeitsmenschen

von *Hansjörg Münster*

Großraumbüros sind heute ja schon der Standard. Dies mag in vielerlei Hinsicht sinnvoll und gut sein - Kosten werden gesenkt, Teamarbeit wird gefördert etc. Aber manchmal muss man sich auch zurückziehen können, insbesondere bei Arbeiten, die eine hohe Konzentration erfordern.



Quelle: Adobe Stock; Urheber: Piumadaquila

Dies fällt im Großraumbüro allerdings oft schwer. Die Globalisierung bringt es dazu auch noch mit sich, dass wir oft mit Kollegen telefonieren, neu hochdeutsch also „skypen" müssen. Damit erhöht sich der Geräuschpegel wieder und stört.

Doch Panasonic hat dafür eine spannende Lösung vorgestellt: Scheuklappen für die modernen Arbeitsmenschen: Sie funktionieren ganz gleich wie die Scheuklappen unserer vierbeinigen Arbeits- und Reittiere: Klappen rechts und links neben den Augen schränken das Sichtfeld ein, sodass nur mehr der Arbeitsbereich und der Monitor sichtbar sind. Es sind jedoch auch Kopfhörer eingebaut und dadurch unterscheiden sie sich doch erheblich von den einfachen Scheuklappen der Pferde.

Und weil wir in einer High-Tech Welt leben, sind es natürlich Noise-Cancelling Kopfhörer, die die Umgebungsgeräusche herausfiltern.

Wir würden nicht im Heute leben, wenn diese Scheuklappen nicht via Bluetooth angebunden wären und nicht über eine eigene App gesteuert werden könnten. Nur der Name für diese Erfindung erschließt sich mir noch nicht ganz: „Wear Space".

Ich weiß nicht, ob sich „Wear Space" bei uns durchsetzen wird. Mir persönlich wäre lieber, wenn in den Großraumbüros eine Möglichkeit gegeben wäre, sich auch einmal zurück zu ziehen. Architekten bzw. Büroausstatter haben hier schon einige Vorschläge am Start. Bevor wir uns als brave Arbeitstiere Scheuklappen aufsetzen, auch wenn die noch so smart, high-tech und hip sind. ■

Links:
https://futurezone.at/produkte/panasonic-bringt-scheuklappen-fuer-buero-angestellte/400148676

https://www.golem.de/news/wear-space-panasonic-baut-scheuklappen-fuers-buero-1810-137183.html

https://derstandard.at/2000089601624/Scheuklappen-fuer-Menschen-um-Grossraumbueros-ertraeglich-zu-machen

Hansjörg Münster ist Principal Consultant und Teamlead bei SEQIS.

Als Allrounder deckt er ein breites Spektrum an Aufgaben ab. Die Schwerpunkte seiner Tätigkeit liegen in den Bereichen Test Management, Testautomation und Lasttest.

Ganz oben auf der Prioritätenliste des IT Profis steht, einen Nutzen und Mehrwert in der Qualitätssicherung seiner IT Projekte zu generieren.

Toolchain in der IT-Analyse

von *Josef Falk*

Für Softwareentwicklung braucht man Werkzeuge – klar. Zunächst muss man sein Programm irgendwo schreiben, man benötigt also so etwas wie einen Editor. Dann muss das Programm kompiliert werden, damit es ausgeführt werden kann. Und damit ist es noch lange nicht getan. Man braucht Bibliotheken, um z.B. auf die Funktionen des Betriebssystems zugreifen zu können. Mithilfe eines Debuggers kommt man Fehlern auf die Spur. Weitere Werkzeuge werden für das Deployment und für die Testautomation benötigt.

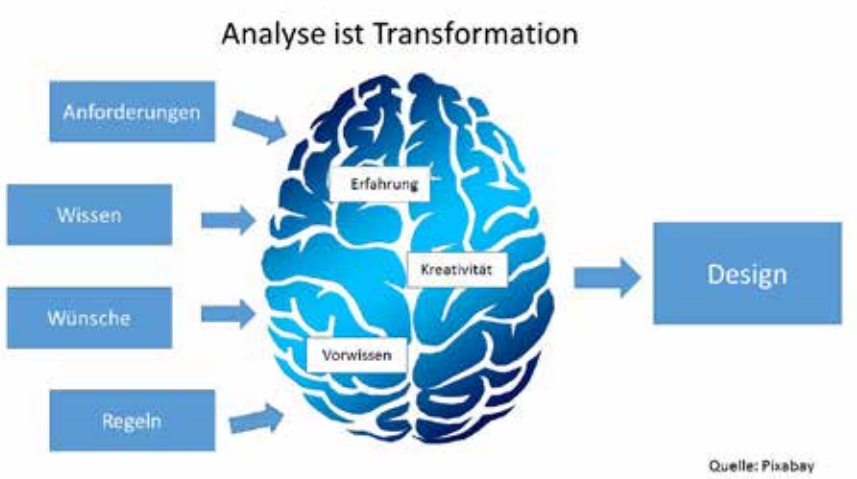


Abb. 1: Abbildung 1: Der Transformationsprozess in der IT-Analyse

Das alles betrifft aber in erster Linie die Entwicklung. In diesem Artikel wollen wir uns die Frage stellen, wie das in der Analyse ist. Benötigen wir auch hier Werkzeuge? Und welche?

Wir wollen uns dieser Frage in folgenden Schritten nähern:

1. Um zu wissen, welche Werkzeuge wir benötigen, müssen wir zunächst klären, was eigentlich die Aufgabe der Analyse ist.
2. Im zweiten Schritt beschreiben wir das wichtigste Werkzeug des Analytikers.
3. Schließlich werden wir mehrere das Haupt-Werkzeug unterstützender Werkzeuge kennenlernen.

Die Aufgabe der IT-Analyse

Um beurteilen zu können, welche Werkzeuge in der IT-Analyse benötigt werden, ist zunächst zu klären, was überhaupt deren Aufgabe ist.

Die Aufgaben in einem IT-Projekt lassen sich in drei Gruppen teilen:

- *Ingenieure:* sie erfüllen all jene Aufgaben, für die man (software-)technisches Wissen benötigt. Jemand muss die Programmiersprache beherrschen, wissen, wie man eine Datenbank aufsetzt, alle die eingangs erwähnten Werkzeuge beherrschen.
- *Manager*: diese kümmern sich darum, dass das IT-Projekt mit den vorhandenen Ressourcen, Zeit und Budget, abgewickelt wird.
- *Gestalter*: deren Aufgabe ist es, dafür zu sorgen, dass das geplante System die ihm zugedachte Aufgabe erfüllt.

Es ist offensichtlich, dass die IT-Analyse die Rolle der Gestaltung übernimmt. Es ist also die Aufgabe der IT-Analyse, das neue System zu gestalten. Basis dafür ist Wissen über das Fachgebiet, für das das IT-System geplant ist. Dieses Wissen erhält der Analytiker aus verschiedenen Quellen, z. B. aus Büchern, diversen Unterlagen, und vor allem aus Gesprächen mit den sogenannten Stakeholdern.
Die Aufgabe des Analytikers ist es also, Wissen (bzw. Anforderungen) in das Design des geplanten Systems zu transformieren.

Das Haupt-Werkzeug am Web von den Anforderungen zum IT-System

Für diese Transformationsaufgabe braucht der IT-Analytiker zunächst kein Werkzeug. Oder aber – wenn man so will – er braucht genau ein Werkzeug: sein/ihr Gehirn.
Das Gehirn, das ist der Ort, an dem sich dieser Transformationsprozess vollzieht. Kreativität, Erfahrung, Gestaltungskraft machen aus Anforderungen ein Lösungsdesign. Diese Kernaufgabe der IT-Analyse funktioniert ganz ohne (weitere) Werkzeugunterstützung.

Heißt das also, in der IT-Analyse benötigen wir gar keine Werkzeuge – reicht uns unser Gehirn, das wir sowieso immer dabei haben? In der Theorie könnte man es so sehen. In der Realität stimmt das aber nicht – und das aus folgenden Gründen:

- *Komplexität:* ein typisches IT-Design ist sehr umfangreich, sodass es ein IT-Analytiker nicht schafft, sich jedes Detail, das er sich bereits überlegt hat, im Gedächtnis zu halten. Der

Ohne Titel, Künstler: Andrea Winkelmayr, Technik: Buntstiftzeichnung

Zwischenstand des Transformationsprozesses muss festgehalten werden, damit später wieder darauf aufgesetzt werden und dieser weitergeführt werden kann.

- *Kommunikation:* der Analytiker kommuniziert sein Design in zwei Richtungen:
  - > *Fachbereich:* die Auftraggeber bzw. die künftigen Anwender müssen das vom Analytiker entwickelte Design akzeptieren. Dazu muss der Analytiker in der Lage sein, das Design zu präsentieren. Dafür ist es wiederum erforderlich, dass es in irgendeiner Form aufgeschrieben wird.
  - > *IT-Technik:* das Design allein bringt noch gar keinen Nutzen; den bringt es erst, wenn es in Software umgesetzt wird. Damit die Entwicklung dazu in der Lage ist, muss dieses Design in geeigneter Form niedergeschrieben sein.
- *Dokumentation:* und schließlich soll der Entwurf des Analytikers auch für die Zukunft bewahrt werden. Das Software-System – sobald es realisiert ist – muss gewartet werden, es müssen Fehler korrigiert werden und auch eine Weiterentwicklung ist in praktisch allen Fällen erforderlich. Dokumentation ist dafür zwingend erforderlich.

Das heißt also, es reicht nicht aus, dass der Entwurf des IT-Analytikers in dessen Gehirn existiert. Er muss in geeigneter Form festgehalten werden – und dafür sind Werkzeuge erforderlich. Welche das sind, das soll im Folgenden ausgeführt werden.

Elemente des IT-Designs

Was also benötigt man, um ein IT-Design zu beschreiben? Diese Frage lässt sich nur beantworten, wenn man sich darüber im Klaren ist, woraus ein derartiger Entwurf besteht. Es gibt ja die unterschiedlichsten Arten von IT-Systemen. Gibt es ein durchgängiges Muster, das die Lösungen all dieser Systeme beschreiben kann?

Jedes IT-System, egal welcher Art, hat drei Aspekte:

- *Statischer Aspekt:* Das ist die Basis jedes IT-Systems. Es beschreibt die Daten, die verwaltet und gespeichert werden müssen, um die Realität des Fachbereiches abzubilden.
- *Dynamischer Aspekt:* Beschreibt die Prozesse und Algorithmen, die die Daten, die im statischen Aspekt definiert werden, erzeugen, transformieren und dem Benutzer zur Verfügung stellen.
- *Schnittstellen:* Es wird beschrieben, auf welchem Weg Daten ins System kommen und weitergegeben werden.

Werkzeuge zur Beschreibung des IT-Designs

Auch wenn die Entstehung des IT-Designs ein geistiger, kreativer Prozess ist, für den eine Werkzeugunterstützung nur beschränkt möglich ist, benötigt es für Kommunikation und Dokumentation Werkzeuge.
Das Design kann auf zwei Arten beschrieben werden:

- *Text:* Beschreibung des Designs durch natürliche Sprache
- *Modelle:* Verwendung von Modellierung zur Beschreibung des Designs.

Textuelle Beschreibung

Natürlich kann ein Design durch einfache Textverwaltungsprogramme beschrieben werden. Sie haben die Vorteile, dass sie universell verfügbar sind und die Integration von Text, Tabellen und Diagrammen sehr gut unterstützen. Diese Tools stoßen aber vor allem bei größeren Teams schnell an ihre Grenzen, insbesondere was die Mehrbenutzerfähigkeit betrifft. Auch die Verwaltung von sehr großen Dokumenten ist in einfachen Textwerkzeugen schwierig.

Eine gute Alternative zu diesen Textwerkzeugen ist die Verwendung von Kollaborationstools oder Wikis. Sie zeichnen sich durch gute Mehrbenutzerfähigkeit und Such- und Navigiermöglichkeiten aus. Häufig sind derartige Werkzeuge auch mit einem Issue-Tracker integriert. Nachteile von Kollaborationstools gegenüber einfachen Textwerkzeugen sind deren mangelnde Offline-Fähigkeit. Auch die Integration von Diagrammen ist manchmal aufwendig.

Modellierung

Ohne Modellierung ist die Entwicklung und Darstellung eines IT-Designs schwer vorstellbar. Etwa

ein Datenmodell ausschließlich in Worten zu beschreiben, ist kaum denkbar. Als wichtigste Methode für die Software-Modellierung hat sich die UML (Unified Modeling Language) durchgesetzt. Sie stellt insgesamt 14 Diagrammarten zur Verfügung. Von diesen werden jedoch in der Regel nicht mehr als fünf benötigt. Daneben gibt es andere Methoden, wie z.B. BPMN (Business Model and Notation) für die Prozessmodellierung.
Welche Methoden sind nun wichtig für die IT-Analyse. Wir wollen dafür auf die Elemente der oben beschriebenen Elemente des IT-Designs zurückkommen.

Statische Analyse
In der statischen Analyse wird gewissermaßen das Skelett des IT-Systems definiert. Dabei wird ein fachliches Datenmodell erstellt. Wichtigste Methode dafür sind die UML-Klassendiagramme.
Gelegentlich kommen auch noch Entity-Relationship-Diagramme zum Einsatz.

Dynamische Analyse
Durch die dynamische Analyse kommt Leben in das statische Design. Der Analytiker beschreibt zu diesem Zweck Algorithmen und Prozesse. Die geeigneten Methoden dafür sind:

- UML-Aktivitätsdiagramme
- Andere Prozessdiagramme: z.B. BPMN
- UML-Sequenzdiagramme
- UML-Statusübergangs-diagramme

Schnittstellen
Durch die Definition von Schnitt-stellen wird beschrieben, wie sich das System nach außen darstellt. Es ist zu unterscheiden zwischen der Schnitt-stelle zum Menschen hin und jener zu anderen Systemen. Die Schnittstelle zum Menschen wird meist durch die Beschreibung von GUI's (Graphical User Interface) festgelegt.
Schnittstellen zu anderen Systemen werden durch API-Schnittstellen definiert.

... und die Werkzeuge
UML- und andere Diagramme können entweder mit einfachen Zeichen-werkzeugen erstellt werden oder man verwendet dafür spezielle Modellierungstools.
Zeichenwerkzeuge liefern optisch attraktive Resultate und bieten hohe Flexibilität. Allerdings muss man sich dabei bewusst sein, dass man zeichnet und nicht modelliert. Eine formal exakte Modellierung ist mit derartigen Werkzeugen nicht möglich.

Modellierungswerkzeuge hingegen erlauben eine effiziente Erstellung und Pflege von Diagrammen und Modellen. Sie sind jedoch eher schlecht geeignet für Text und Tabellen.

Der Entwurf von Schnittstellen unterscheidet sich je nachdem, ob es sich um Schnittstellen zu einem menschlichen Anwender oder zu einem anderen System handelt. Der Entwurf von Screens, die als Schnittstelle zum Menschen hin dienen, kann durch ein Mockup- oder Wireframe-Werkzeug unterstützt werden. Das Design von API-Schnittstellen zu anderen Systemen wird durch textuelle Beschreibung oder durch Modelle festgehalten.

Zusammenfassung
Das Design, das ein Analytiker gestaltet, entsteht in erster Linie in seinem Kopf. Dieser Entstehungs-prozess benötigt kaum Werkzeuge. Damit dieses Design aber dokumentiert und kommuniziert werden kann, muss es mit Hilfe von Werkzeugen niedergeschrieben werden. Am besten sind dafür geeignet:

- Ein Wiki oder Kollaborationstool für die beschreibenden Texte
- Ein Modellierungswerkzeug zum Erstellen von Modellen, das die verschiedene UML-Diagramme und eventuell weitere Diagram-me, wie BPMN oder Daten-flussdiagramme unterstützt
- Ein Mockup-Tool, mit dessen Hilfe Screens für die GUI-Schnittstelle entworfen werden können.

Mit diesen Tools kann der Analytiker das Produkt des Transformationsprozesses, der aus Anforderungen ein IT-Design macht, festhalten und so die Basis für eine erfolgreiche Umsetzung legen. ■

Literatur:
Starke, Gernot: Effektive Softwarearchitekturen, 7. Auflage, 2015 Carl Hanser Verlag München

Mag. Josef Falk ist IT Analytiker.

Seit dem Abschluss seines Studiums der Betriebswirtschaftslehre in Wien gestaltet er Lösungen in den unterschiedlichsten Fach-bereichen – und ist dabei Mittler zwischen Fachbereich und IT-Entwicklung.

Besonderes Augenmerk legt er bei der Analyse auf den Innovationsgrad. Neben seiner Projekttätigkeit befasst er sich mit der Entwicklung der Business Analyse und ist aktuell Mitglied des Vorstandes des Austria Chapter des IIBA (International Institute of Business Analysis).

Toolchain: Einer um uns alle zu knechten?

von *Klemens Loschy*

Sie werden es vielleicht bereits ahnen, welches Tool ich meine: das Konvolut aus Atlassian Jira Confluence. In unserem Projektalltag sehen wir kaum ein (IT) Unternehmen, das noch nicht (zu welchen Teilen auch immer) auf dieses dynamische Duo setzt - insofern darf ein Artikel dazu natürlich nicht fehlen.

Dabei sind die Anwendungsgebiete teils wirklich überraschend: Von der Grundidee, Issues (oder wer es in Deutsch nutzt: Vorgänge) zu verwalten ist man dabei meistens bereits meilenweit entfernt. Durch die (äußerst) komplexe und (überbordend) umfangreiche Möglichkeit ein Jira Projekt anzupassen ist es defakto möglich, *alles* in einem Jira Projekt abzubilden. Da findet man neben den (mittlerweile) Standard Einsatzgebieten wie das Management kompletter agiler Projekte, oder der Verwaltung einfacher ToDo-Listen auch etwas kuriosere Konfigurationen wie die Abbildung des Jahresabschlusses der Finanzabteilung oder die Sammlung und Auswertung von Unternehmens-Gamifications.

Als SEQIS-Mitarbeiter versuchen wir immer die vorhandenen Produkte unserer Kunden zu berücksichtigen: passt das aktuell eingesetzte Tool noch ins Zielbild, welche Gaps, Probleme oder Workarounds werden nötig sein und wie verhält sich das zu einem Umstieg auf ein oder mehrere neue Tools? Zumeist bleibt man bei der vorhandenen Toollandschaft, das ist dann oft Atlassian. Unter uns: haben Sie schon mal das lang verwendete Tool X durch Tool Y in Ihrem Unternehmen abgelöst? Solche Vorhaben führen fast immer zu sehr spannenden Projekten! Also bleiben wir bei den Atlassian Produkten, die können ja eh alles... mehr oder weniger. Durch das Tester Auge betrachtet kann Jira in Wahrheit gar nicht so viel.

Klar, man kann Jira schon so herrichten, dass alle testrelevanten Artefakte und Informationen

Quelle: Fotolia; Urheber: Sergey Nivens

persistiert und analysiert werden können, nur das führt entweder zu 1) erheblichen Abstrichen bei der Benutzbarkeit, Intuivität und auch in puncto Funktionsumfang oder 2) zu kostenpflichtigen Add-Ons (Zephyr und XRay seien hier genannt) und nicht ganz so hohen Abstrichen der Punkte unter 1).

Wie man es dreht und wendet, Jira ist aus Test und Testmanagement Sicht sicher nicht das Tool ganz oben auf der Liste der potentiellen Kandidaten (nicht, dass die Liste so lange wäre...). Die Ursache ist auch einfach zu nennen: Jira ist und bleibt ein Ticketing Tool! Testen und Test-management bedingt eben mehr als das: unter anderem sollen parametrisierbare Komponenten zu Testfällen verknüpft werden, Anforderungen abgedeckt, Testfälle zu Test-Sets gruppiert, Testsessions geplant, durchgeführt und dokumentiert, Abweichungen erfasst und der aktuelle Qualitätszustand über alle Teststufen jederzeit einfach dargestellt werden können. Und auch wenn einiges davon durch die beiden genannten AddOns möglich ist, es fühlt sich trotzdem nicht gut an.

Spätestens jetzt stellt sich die Frage: wollen wir uns und unsere Arbeits-weise an ein Tool anpassen oder soll sich die Toolchain an unsere Anforderungen anpassen? Bei dieser Fragestellung werden fast alle mit „natürlich muss sich die Toolchain anpassen" antworten, aber wieso tut sie das im Regelfall dann doch nicht? Atlassian hat das schon ganz geschickt gemacht: der Einstieg in die Produkte ist schnell, einfach und günstig. Und ehe man sich versieht, will man sie nicht mehr hergeben. Dann beginnt man mit AddOns und Customisierungen die wachsenden Anforderungen doch noch irgend-wie ins Jira hineinzubekommen, und letztendlich ist man genau dort: wir passen uns an das Tool an und nicht mehr umgekehrt.

Was möchte ich mit dem Artikel erreichen? Keinesfalls möchte ich die Atlassian Produkte verunglimpfen. Ich möchte aber zum Umdenken anstoßen. Begeben wir uns doch mutig auf die Suche nach Tools, die wirklich unsere Anforderungen abdecken, die wirklich die Lösung unserer Aufgabenstellung implementieren und nicht bloß „auch" können. Blicken wir wieder über den Tellerrand und geben anderen Tools eine faire Chance. Sehr viele (Test & Testmanagement) Tools, wenn nicht fast alle, sprechen „Atlassisch" und lassen sich ohne weiteres in Jira und Confluence nahtlos integrieren.

In dem Sinne: Passen wir die Toolchain unseren Bedürfnissen an und nicht (mehr) umgekehrt! ■

Links:

https://de.atlassian.com/
https://de.atlassian.com/software/ jira

Klemens Loschy ist Principal Consultant, Teamlead und Spezialist in den Bereichen IT-Analyse, Software Test und Projekt Management.

Er kann auf jahrelange Erfahrung in den Bereichen Testautomation, Last-Tests und Performance Engineering, funktionale Tests, Testen in agilen Teams, Anwendungs-entwicklung von Testsoftware sowie Beratung und Unterstützung in zahlreichen Projekten unterschiedlichster Branchen zurückblicken.

SEQIS Kunden im Rampenlicht: OMV

von *Hansjörg Münster/Christina Eder*

SEQIS hat dank strukturierter und detaillierter Projekt- & CutOff-Planung die reibungslose Migration unseres IDMs ermöglicht. Durch die agile und transparente Projektleitung und die Qualitätssicherung der neuen Version, der Integration in unsere Sytemlandschaft und die Absicherung der Migration konnte die Umstellung der Systeme ohne Probleme und Einbußen im Tagesgeschäft umgesetzt werden.

Thomas Laggner, Head of Workplace Management

Qualitätssicherung und Projektleitung der Migration des Identity Management Systems („IDM")

End of Support für die aktuell verwendete IDM Version und der Wunsch aktuelle Features nutzbar zu machen, machten das Upgrade notwendig. SEQIS übernahm die Aufgaben der Qualitätssicherung und des Projektmanagements und konnte so gemeinsam mit OMV und dem externen technischen IDM Dienstleister die Migration erfolgreich abschließen.

Durch das Erstellen und Anpassen von Use Cases und Test Cases war eine reibungslose und strukturierte Überprüfung der IDM Funktionen in der neuen Release möglich – dadurch wurde der Arbeitsalltag der Mitarbeiter und das Tagesgeschäft durch den Umstieg nicht beeinträchtigt.

IDM Migration: Besondere Herausforderungen

Die Migration von komplexen und stark vernetzten IT-Systemen, wie das IDM, ist immer eine heikle Sache. Es muss sichergestellt sein, dass die notwendigen Funktionalitäten auch nach dem Upgrade vorhanden sind, dass die Integration weiterhin gegeben ist und dass Vollständigkeit und Integrität der migrierten Daten gewährleistet sind. Zudem gibt es in einem Migrationsprojekt immer Zeitdruck: In der Regel steht für die Migration selbst nur eine eingeschränkte Zeitspanne, wie beispielsweise hier nur ein einzelnes Wochenende, zur Verfügung.

Die oben genannten sind ein paar Gründe die dafür sprechen sich mit Respekt dieser Aufgabe zu nähern und gegebenenfalls professionelle externe Unterstützung heranzuziehen. Ein Muss für solche Projekte ist einerseits eine verantwortungsbewusste, vorausschauende und transparente Projektplanung und auch eine lückenlose und aktuelle Dokumentation der verwendeten Funktionen vor der Migration. Beides bedingt im Regelfall (wie auch hier) eine intensive Analysephase:
Im Rahmen der Vorbereitung wurden seitens SEQIS folgende Aspekte analysiert und getestet:

- Wie ist der allgemeine Qualitätszustand des neuen IDM Ziel-Releases? Oft will man beim Upgrade ja auch gleich die „aktuellste" Version verwenden...
- Wie ist der funktionale Umfang des neuen Realeases, gibt es auch immer noch alle bisher verwendeten Funktionen, oder wurden diese durch neue Funktionen ersetzt/abgelöst?
- Wie kann die Migration selbst abgesichert werden? Macht es Sinn, neben den Anleitungen des Herstellers das Testportfolio zu

Die Aufgabe

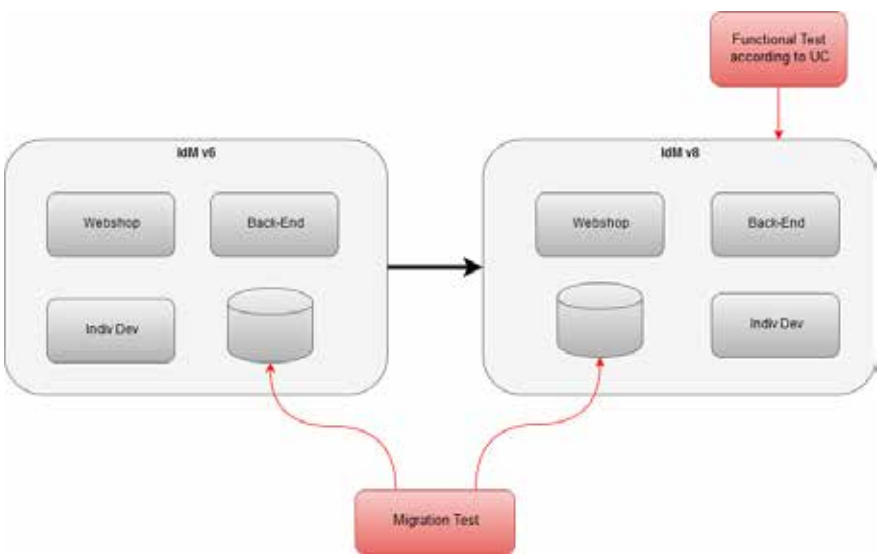
- *Kunde:* OMV
- *Arbeitsauftrag:* Qualitätssicherung und Projektleitung der Migration von One Identity Management (IDM) von Version 6 auf Version 8
- *Umfang*: 7 Monate (Dezember 2017 bis Juni 2018)
- Manpower: 2 FTE, aufgeteilt auf 3 Consultants
- *Tools*: Atlassian's JIRA & Confluence (Projektmanagement, Dokumentation und Testmanagement)

Die Lösung

- Erfolgreiche Migration von IDM von Version 6 auf Version 8 durch begleitende Qualitätssicherung (Test)
- Erfolgreiche Projektleitung für das Migrationsprojekt
- Einführung und Vorleben von agilen Methoden
- Dokumentation und Videoanleitung für End User und Admins
- Use Case-Dokumentation mit Ablaufdiagrammen
- Testfallerstellung und -durchführung

erweitern?
- Wie sieht es mit einer allgemeinen Bereinigung von Code, Daten, Workflows,... vor der eigentlichen Migration aus? Gibt es ungenutzte Teile am Startpunkt?
- Wie sind die Schnittstellen und die Integration zu anderen verbundenen Systemen? Wurden diese mit den der neuen Release entsprechenden Funktionen angebunden?

Um eine transparente Projektplanung und durchgängige Dokumentation von Use Cases, Requirements und Tests zu etablieren hat SEQIS in diesem Fall kurzfristig eine cloud-basierte Toolchain mit Atlassian's JIRA und Confluence aufgebaut. Damit war der Projektfortschritt zu jeder Zeit sichtbar und die Planung konnte in Abstimmung mit allen Stakeholdern immer weiter detailliert werden. Nicht zuletzt ist auch der methodische Software Test enorm wichtig, denn am Ende des Tages kommt es darauf an, wie gut die neue Version nach der Migration funktioniert. Es wurden folgende Tests durchgeführt:



Funktionale Tests der neuen Version
Integrations- und Schnittstellentests
Test der Migration
Damit im Tagesgeschehen auch alle Benutzer des IDM Webshops mit dem Upgrade arbeiten konnten, war es unverzichtbar, neben der Dokumentation zu den Anforderungen und Tests, auch die Benutzerdokumentation zu aktualisieren. Zusätzlich wurden, zur Vereinfachung für die Benutzer, für die am häufigsten benutzten Arbeitsabläufe Anleitungen als Videos bereitgestellt.

IDM Migration: Umstiegspläne optimieren und testen

Die Migration von IDM Systemen bringt natürlich viele spezielle Risiken und Gefahrenpotenziale, die vor der Umstellung mitigiert werden müssen. Viele dieser Probleme kann man durch entsprechendes Projektmanagement, aber auch durch einen methodischen Softwaretest adressieren. Genau in diesen Segmenten ist SEQIS spezialisiert und konnte damit mit dem richtigen Know-How sehr schnell in

Bild: OMV; Quelle: OMV

Rubrik „Im Spotlight":

Unsere Kunden stehen bei all unseren Aktivitäten im Fokus. Egal welche Branche, egal welche Technik, egal welche Aufgabenstellung: wir beraten und unterstützen gerne.

In der Rubrik Spotlight stellen wir regelmäßig unsere Kunden auch in der SEQIS QualityNews in den Vordergrund und geben dadurch einen Eiblick in unsere unterschiedlichen Projekte und Dienstleistungen.

das Projekt einsteigen und mit einer detaillierten Planung den Erfolg des Projekts allgemein und des Umstiegs absichern.

SEQIS arbeitet seit langem mit agilen Methoden, die die Umsetzung des Arbeitsauftrags effektiver und effizienter gestalten und eine schnelle Reaktion auf Änderungen zulassen. Diese dienen einerseits dazu, den Status der einzelnen Planungs- und Umsetzungsschritte transparent zu halten und helfen andererseits auch dem Verständnis zwischen internen und externen Projektmitarbeitern und der Kommunikation mit Auftraggebern.

Wie in vielen anderen Lebensbereichen sind auch in der IT Projektleitung Kommunikation und Vertrauen zwischen den Akteuren zumindest die halbe Miete und der Grundsatz für einen reibungslosen Projektablauf.

IDM Migration: Der Kunde OMV

Die OMV fördert und vermarktet Öl und Gas, innovative Energielösungen und hochwertige petrochemische Produkte.

Mit einem Konzernumsatz von EUR 20 Mrd und einem Mitarbeiterstand von rund 20.700 im Jahr 2017 ist die OMV Aktiengesellschaft eines der größten börsennotierten Industrieunternehmen Österreichs.

Gegründet wurde OMV am 3. Juli 1956 als österreichische Mineralölverwaltung Aktiengesellschaft (ÖMV). 1960 ging bereits die Raffinerie Schwechat in Betrieb. Ende der 60er Jahre wurde Erdgas über einen Liefervertrag mit der damaligen UdSSR mit in den Vertrieb aufgenommen. 1991 stieg die OMV in das internationale Tankstellengeschäft im südosteuropäischen Raum ein, 2006 kam die Türkei dazu. Die Marke VIVA als Startschuss des heute bedeutenden Non-Oil-Business in der OMV wurde 1994 mit dem ersten VIVA Shop in Oberwart ins Leben gerufen. 1995 wurde die damals noch ÖMV in OMV umbenannt und vollzog so den Schritt zum internationalen Unternehmen.

Die OMV betreibt heute mehr als 2.000 Tankstellen in zehn Ländern und verfügt über Gasspeicher in Österreich sowie Deutschland. Sie leistet einen wesentlichen Beitrag zur Energieversorgung Österreichs und ihrer Kernmärkte mit rund 200 Milllionen Kundinnen und Kunden in Mitteleuropa, Südosteuropa sowie der Türkei.■

Thomas Laggner ist Head of Workplace Management bei der OMV.

Die Toolchain für den Tester

von *Sabrina Zwaiger*

Durch meine Tätigkeit als Softwaretesterin in traditionellen und auch in agilen Projekten konnte ich die unterschiedlichsten Erfahrungen sammeln. In den verschiedenen Projekten hatte ich immer wieder mit unterschiedlichen Tools und/ oder Programmen zu tun.
Als Vorbereitung für diesen Artikel zum Thema „Toolchain für den Tester" habe ich mich mir folgende Fragen gestellt und möchte euch diese Fragen und meine Antworten hier vorstellen.

Die erste Frage, die mir in den Sinn kam, war folgende:

1) Welche Aufgabe hat ein Softwaretester?

Um zu wissen welche Tools ein Tester benötigt, sollten wir uns zuerst ansehen welche Aufgaben ein Tester hat.

Die Aufgabe eines Testers ist das Finden von Fehlern, das Aufdecken von Abweichungen gegenüber der Spezifikation. Dies hört sich im ersten Moment etwas destruktiv an, jedoch ist Testen eine extrem kreative und herausfordernde Aufgabe. Als Tester sollte mach sich bewusst sein, dass auch die Soft Skills wichtig sind. Tester sollten positiv und lösungsorientiert sein. Sie sollten eine eher kritische, qualitätsorientierte, skeptische Denkweise über das Produkt an den Tag legen. Testergebnisse, Testfortschritte und Produktqualität sollten genau beurteilt werden und man sollte darüber sachlich berichten.
Die Teamfähigkeiten eines Testers sind extrem gefragt, wenn sie zusammen mit Kunden bzw. dem Fachbereich effektiv an der Definition prüfbarer User-Stories bzw. Anforderungen und insbesondere auch an den Abnahmekriterien arbeiten. Die Team- und Kommunikationsfähigkeiten sind ebenfalls erforderlich, wenn sie Fehlerberichte erstellen und mit den Entwicklern oder dem Team bei der Fehlerbehebung zusammenarbeiten.

Folgende Tätigkeiten sind durchzuführen bzw. werden folgende Dinge benötigt damit der Tester seine Aufgaben erledigen und eine Software/ein Objekt testen kann:

- Idealerweise Zugang zu den Anforderungen bzw. zur Spezifikation der Software oder entsprechende Heuristiken
- Er benötigt eine Teststrategie
- Er benötigt eine Testumgebung und Zugang zu dieser
- Testdaten müssen konfiguriert bzw. erstellt werden – kann der Tester die Testdaten selbständig erstellen oder braucht er vielleicht den Kunden dazu?
- Wie und wo werden die Testfälle und die dazugehörende Testdurchführung dokumentiert?
- Möglicherweise Zugang zu Testwerkzeugen, um automatisierte Tests durchführen zu können
- Zugang zum Tool für die Fehlerberichte

Nachdem dieser Punkt geklärt war, stellte ich mir nun folgende Frage:

2) Welche Werkzeuge braucht ein Softwaretester, um seine Aufgabe erledigen zu können?

Meiner Meinung nach sind die wichtigsten Werkzeuge eines Softwaretesters seine Softskills. Ohne seine Kreativität kann er keine Testfälle erstellen. Ohne seine Kommunikationsfähigkeiten kann er weder mit dem Kunden noch mit den Entwicklern kommunizieren. Ohne seine Teamfähigkeit kann er keine Schnittstelle zwischen Kunde und Entwickler sein.

Darüber hinaus braucht er natürlich auch eine Reihe von technischen Werkzeugen. Schauen uns wir diese einmal an:

2.a.) Um die Arbeit starten zu können braucht ein Tester sowie auch der Entwickler Zugang zur Programmspezifikation.
Ein Dokument oder mehrere Dokumente, die die Anforderungen, den Aufbau und das Verhalten des gebauten Systems bzw. der Komponenten beschreibt. Diese dient den Entwicklern als Grundlage für die Programmierung und den Testern als Grundlage für das Ableiten von Testfällen. Manchmal beinhaltet die Spezifikation auch Vorgaben zur Überprüfung/zum Testen der Anforderungen beispielsweise Abnahmekriterien oder in agilen Teams die Definition of Done.

Das Tool, indem die Anforderungen erfasst sind, beinhaltet sehr oft Texte oder sogar Dateien, es werden oft Grafiken für ein Design hinterlegt. In einigen Tools werden auch Flussdiagramme oder Ablaufdiagramme abgespeichert. Diese Diagramme zeigen oft den Ablauf/die Verwendung eines Programmes oder die Veränderung der Daten innerhalb eines Programmes.

2.b.) Wo und wie werden die Testfälle und die Testdurchführung erfasst und dokumentiert?
In vielen Projekten gibt es eine einheitliche Regelung wo und wie

Quelle: Adobe Stock; Urheber: XtravaganT

Testfälle dokumentiert und durchgeführt werden. Wurde ein Testfall durchgeführt, wenn es keine Dokumentation dazu gibt? Ja, wenn ein Fehler dabei aufgetreten ist und es dazu einen Fehlerbericht gibt. Grundsätzlich empfehle ich zu dokumentieren; jedoch kann die Dokumentation auch nur aus einer kurzen Beschreibung des Testfalles bestehen. „Just enough" wie man im Agilen so schön sagt.

Eine Dokumentation kann aus vielen Gründen wichtig sein, z.B.

- Nach einer bestimmten Anzahl an Testfällen weiß man nicht mehr welchen man schon durchgeführt hat und welchen nicht
- Um Testfälle wiederholen zu können bzw. damit jemand anderer auch die Testfälle wiederholen kann/muss
- Ergebnisse der Testdurchführung sind oft die Grundlage für Testberichte an den Kunden
- Es gäbe noch eine Menge an Gründen, die für eine Dokumentation spricht, Fazit ist auf jeden Fall: bitte eine Dokumentation durchführen und in einem gemeinsamen Tool speichern, archivieren und/oder für die Testergebnisprotokolle zur Verfügung stellen.

2.c.) Werden vielleicht neben den manuellen Test auch automatisierte Tests durchgeführt?

- Welches Tool wird für die automatisierten Tests verwendet?
- Was benötige ich als Tester um die automatisierten Tests durchzuführen, zu dokumentieren und/oder gegebenenfalls zu ändern oder zu erweitern?

Vielleicht können die manuellen und die automatisierten Tests im gleichen Tool dokumentiert werden.
Das Thema „automatisierte Tests" kann in manchen Fällen etwas komplexer sein und sollte daher in Projekten genau betrachtet und analysiert werden.

2.d.) Ein weiteres Tool, das der Tester benötigt, ist zur Dokumentation der Fehlerberichte bzw. Fehlermeldungen.
Üblicherweise wird in einem Projekt eine zentrale Fehlerdatenbank eingerichtet, in dieser werden alle Fehlerzustände oder Defekte beschrieben, dokumentiert und verwaltet.

Ein Fehlermanagementtool ist auch ein Kommunikationswerkzeug. Es dient dazu, die gefundenen Fehlerwirkungen und Abweichungen dem Entwickler und/oder dem Testmanager sowie dem Kunden mitzuteilen. Wie dies geschieht, kann für die weitere Zusammenarbeit aller beteiligten förderlich sein oder sie negativ beeinflussen.

Um das Defecttool bedienen zu können benötigt der Tester daher seine Softskills Ein Fehlerbericht sollte unbedingt sachlich und genau beschrieben sein. Durch die Fehlerbeschreibung sollte der Testfall von einer dritten Person erneut durchge-

Quelle: Adobe Stock; Urheber: snyGGG

führt und somit auch das Fehlverhalten erneut produziert werden können. In der Beschreibung sind daher alle notwendigen Daten und Fakten zu erfassen. Rückfragen sollten auch über dieses Tool möglich sein.

Testmanager und Projektmanager sollten über das Fehlermanagementtool jederzeit ein aktuelles und vollständiges Bild über den Status der Fehler verschaffen können, über die Anzahl der Fehler, die Klassifizierung und den Korrekturfortschritt.

2.e.) Beim Schreiben dieses Artikels haben ich mich immer wieder gefragt ob eine Testumgebung auch als Werkzeug für den Tester gesehen wird.
Oder auch die Testdatenbanken, in der Testdaten für die Testdurchführung gespeichert werden.
Ob eine Testumgebung und ihr dazugehörenden Backendsysteme und Datenbanken, die die Produktion spiegeln soll, als Tool gilt? Ich habe diese Frage auch mit meinen Kollegen im Büro kurz diskutiert. Fazit ist, es gibt hier mehrere Meinungen und viele Argumente dafür und dagegen.

Fazit
Zusammenfassend kann ich folgendes sagen: Für mich sind die wichtigsten Tools, die ein Tester haben kann, seine Softskills. Darunter vor allem seine „Kommunikationsfähigkeit".
Ohne Kommunikation kann eine Software weder analysiert noch beschrieben werden. Ohne Kommunikation kann ein Programm weder entwickelt noch an andere Programme angeschlossen werden. Ohne Kommunikation kann der Tester die gefundenen Fehler weder beschreiben noch beheben lassen. Ohne Kommunikation kann ein Team aus Analytikern, Entwicklern und Testern keine Software gestalten und erschaffen. Im Arbeitsalltag eines Testers und bei der Handhabung der einzelnen Tools zur Spezifikation, zur Testdurchführung und zum Fehlermanagement ist die Kommunikation das entscheidende Werkzeug! ■

Literatur

Basiswissen Softwaretest, 5. Auflage, dpunkt.verlag
Foundation Level Extension Syllabus Agile Tester, Herausgegeben durch Austrian Testing Board, German Testing Board e.V. und Swiss Testing Board

Sabrina Zwaiger ist Consultant bei SEQIS und Spezialisting Software Test.

Nach ihrer Ausbildung war sie in der Buchhaltung tätig und konnte dort erste Erfahrungen im Softwaretest sammeln. Dies war auch die Entscheidungsbasis für eine Umorientierung in die IT und den Bereich Softwaretest.

Sie arbeitet gerne und Agilen Teams. Die Kommunikation mit den Stakeholdern, Entwicklern, Fachbereichen zur Ableitung von Testfällen und Geschäftsprozessen und der Wille, Probleme nicht nur aufzuzeigen sondern auch zu lösen ist ihr ein wichtiges Anliegen.

SEQIS „10 things“ Expertentreff #4/2018: Agiles Projektmanagement

von *Christina Eder*

Die letzten „10 things“ 2018 hatten das agile Projektmanagement zum Thema. Principal Consultant Hansjörg Münster verriet dem Publikum 10 Tipps und Tricks um agile Projekte erfolgreich zu leiten.

Agile Softwareentwicklung ist heute nichts Exotisches mehr, sondern vielfach erprobter Standard. Das klassische Projektmanagement ist mit sich selbst organisierenden, eigenverantwortlichen und selbstplanenden Teams konfrontiert.

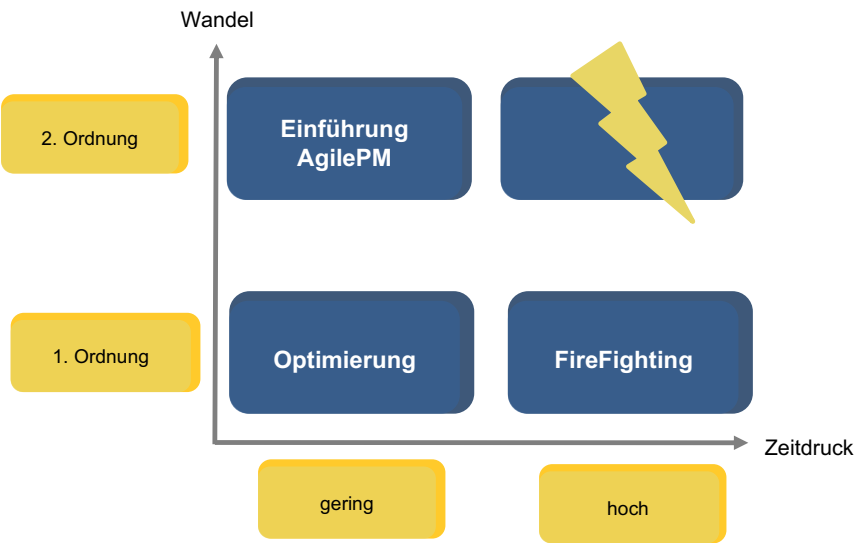
Kurze Planungszyklen (Sprints), Transparenz in den aktuellen Tasks sowie Visualisierung des Fortschrittes sind für etablierte Projektmethoden neu. Kurzfristige Änderungen und Anpassungen der Ziele und Projektinhalte stehen scheinbar im Widerspruch zur klassischen Planung.

Agilität und Lean Management sind heute ohne Widerspruch so erfolgreich, dass sich auch das Projektmanagement anpassen muss.

Die Fragen, die sich stellen, um das Projektmanagement um agile Methoden erfolgreich zu erweitern sind:

- Wie lassen sich klassische und iterative Planung kombinieren?
- Wie kann man im Projekt das Budget steuern, wenn selbstbestimmende Teams den Skope agil ändern können?
- Wie kann man dem Management den Fortschritt reporten, wenn die Inhalte und Ziele in jedem Sprint veränderbar sind?

Hansjörg Münsters 10 Tipps und Tricks helfen, diese Fragen zu beantworten:



1. Wählen Sie den Zeitpunkt der Einführung von AgilePM* mit Bedacht!

* Agile PM heißt abgekürzt „agiles Projektmanagement“. Das bedeutet, dass nicht, wie beim Wasserfallmodell, das gesamte Projekt chronologisch Schritt für Schritt von der Analyse bis zum Code umgesetzt wird. Beim agilen Projektmanagement werden Rahmenbedingungen für kürzere und flexiblere Zyklen geschaffen. Die Prozesse finden parallel statt, damit das den aktuellen Anforderungen entspricht, die sich im Laufe des Projektes oft verändern.

2. Kümmern Sie sich um die Voraussetzungen für ein erstes Projekt!

Die Kernfrage ist: sind wir bereit für agiles Projektmanagement? Um das herauszufinden, stellt das Agile Business Consortium einen Fragebogen zur Ermittlung des Reifegrades zur Verfügung. Den Project Approach Questionnaire erhalten Sie auf der Website www.agilebusiness.org.

3. Bevorzugen Sie bei der Projektmitgliederauswahl Personen mit sozialer Kompetenz und mit Commitment!

Unterstützen Sie das Teambuilding aktiv.

Das Management sollte miteinbezogen werden. Das Commitment „von oben“ ist ein wichtiger Faktor für ein erfolgreiches agiles Projekt. Gleichzeitig ist nicht nur die fachliche, sondern auch die soziale Komponente der Teammitglieder entscheidend. Es kann möglich sein, dass – um diesen Prozess zu unterstützen – ein externer Coach hilfreich ist.

4. Stellen Sie sicher, dass die acht Grundprinzipien allen Projektmitgliedern bekannt sind und achten Sie darauf, dass diese gelebt werden!

Die acht Grundprinzipien des AgilePM lauten:

1. Konzentrieren Sie sich auf das Geschäftsbedürfnis
2. Liefern Sie pünktlich
3. Arbeiten Sie zusammen
4. Dulden Sie keine Abstriche in Sachen Qualität

5. Bauen Sie schrittweise auf soliden Grundlagen auf
6. Entwickeln Sie iterativ
7. Kommunizieren Sie kontinuierlich und deutlich
8. Demonstrieren Sie Steuerung

5. Folgen Sie stets den Grundsätzen des AgilePM!

Das bedeutet: während Zeit, Kosten und Qualität fix sind, ist der Umfang variabel.

6. Liefern Sie pünktlich nach MoSCoW!

Beachten Sie dabei die 60/20-Regel: 60 % Musts, 20 % Coulds.

- *Must: Anforderungen,* die essentiell und nicht verhandelbar sind. Ein „Nicht-Erreichen" würde das Scheitern des Projektes/ Inkrementes/Timebox (Minimum Usable Subset) bedeuten
- *Should:* Anforderungen, die eine hohe Relevanz haben, die wichtig, aber nicht erfolgskritisch sind
- *Could:* Anforderungen, die als „Wünsche" klassifiziert werden, die „Nice to haves". Auswirkungen bei Nicht-Erbringung sind deutlich niedriger
- *Won't:* Was dann noch übrig bleibt, wird nicht umgesetzt

7. Achten Sie stets auf die Qualität der Lieferungen und der verwendeten Prozesse!

Dulden Sie keine Abstriche bei der Qualität!
Qualität wird in zwei verschiedene Bereiche unterteilt: einerseits ist es wichtig, die Lösungsqualität zu betrachten und sich folgende Fragen zu stellen: Wurden die definierten Standards erreicht (Fit for purpose)? Wann ist ein Produkt „gut genug", wie gut erfüllt eine Lösung das Geschäftsbedürfnis?

Das Qualitätsniveau wird bereits in der Foundationphase vereinbart, Abnahmekriterien werden für jede Anforderung extra definiert. Es gibt keine Herabstufung der Qualität, um Kosten oder Deadlines einzuhalten – dies widerspricht dem grundsätzlichen Prinzip des agilen Projektmanagements.
Neben der Lösungsqualität gibt es noch die Prozessqualität als wichtigen Faktor: das AgilePM ist der Rahmen, der auf die jeweiligen Bedürfnisse des Projekts angepasst werden kann. Der Prozess stellt sicher, dass: die richtigen Aktivitäten zum richtigen Zeitpunkt gesetzt werden, Auslassungen und Versehen verhindert werden, Erkenntnisse aus vorangegangenen Erfahrungen eingebracht werden. Dadurch wird die pünktliche und budgetgerechte Lieferung einer (vorhersagbaren) Lösung ermöglicht.

8. Stellen Sie sicher, dass Erkenntnisse zur Verbesserung des Prozesses gewonnen UND in späteren Projekten berücksichtigt werden!

AgilePM ist ein Prozessrahmen, der den eigenen Bedürfnissen angepasst werden und kontinuierlich

verbessert werden soll. Dafür sind regelmäßige Qualityreviews vorgesehen. Diese Retrospektiven dienen der kontinuierlichen Verbesserung des gelebten Prozesses. So beinhaltet z.B. der „Project Review Report" am Ende jedes Projekts ein Protokoll des Erreichten, Erkenntnisse für spätere Inkremente/Projekte und das Ergebnis der Retrospektive.

9. Kommunizieren Sie kontinuierlich und deutlich. Schaffen Sie Ehrlichkeit und Transparenz im Projekt.
Als Ursache für fehlgeschlagene Projekte wird oft unzureichende Kommunikation genannt. Daher stellt AgilePM eine Reihe von Praktiken zur Verbesserung der Kommunikation zur Verfügung: neben den bekannten Daily Standups sind dies z.B. „facilitated Workshops".
Wichtig ist Transparenz und Ehrlichkeit in der Kommunikation um alle Beteiligten am Laufenden zu halten. Darum ist schlanke und aktuelle Dokumentation wesentlich – nur so wird sie auch gelesen. Überdies spielt der Projektleiter eine große Rolle: die Aufgabe dieser Person ist es, die Stakeholder einzubeziehen und über den Projektverlauf zu informieren.

10. Planen Sie Umfang inkrementell unter Berücksichtigung von Zeit, Kosten und Qualität!
Im agilen Projektmanagement ist es wichtig, immer wieder abzugleichen: wie schreitet das Projekt voran in puncto Zeit, Kosten und Qualität? Anpassungen finden laufend statt und nicht nur am Anfang (Plan) und am Ende (Ziel). Und vergessen Sie nie: Dulden Sie keine Abstriche bei der Qualität! ■

Auf www.seqis.com/youtube finden Sie das Interview zur Veranstaltung mit Hansjörg Münster!

MMag. Christina Eder ist Marketing Managerin bei SEQIS. Sie ist erste Ansprechpartnerin für Presse- und Marketinginformationen.

Von Drucksortengestaltung über Video-dreh und -schnitt, klassischer Pressearbeit, Betreuung der SEQIS Onlinekanäle bis hin zur Organisation von Veranstaltungen übernimmt sie Marketing- und Kommunikationsagenden.

Besonders wichtig ist ihr die stetige Entwicklung von Medien, Tools und Konzepten.

KickOff Workshop für die Erstellung von Toolchains

von *Alexander Weichselberger*

Bei der Definition, WAS eine Toolchain leisten muss, empfiehlt sich eine strukturierte Aufnahme der Anforderungen. Wir haben in den letzten Jahren dutzende Toolchains realisiert – folgende unsere Basisempfehlungen für diesen Startpunkt zur „Toolchain als Lösung".

Gehen wir davon aus, dass die Toolauswahl bereits getroffen wurde und auch das Ziel der Realisierung zumindest in seinen groben Zügen vorliegt („Anforderungen"). Darüber hinaus ist auch klar, wer im Fachbereich für diese Realisierung Inputgeber ist. Letzteres ist maßgeblich für die Durchführung eines Kick-Off Workshops mit dem Ziel, folgende Vorstellungen abzustimmen:

- Agenda
- Timeline
- Ziele
- Prinzipien
- Player & Stakeholder
- Knowhow Transfer
- Toolrahmenbedingungen
- MoSCoW
- Toolchain Projekt
- Ideenboard – Jetzt & Später
- Artefakte – Tickettypen und Co
- Funktionen
- Workflow

Sie fragen sich sicherlich, ob es für diese Vielzahl an Punkten eine spezifische Reihenfolge gibt.

Empfehlung Nr 1.: Veranstalten Sie diesen KickOff als WorldCafe ® (https://de.wikipedia.org/wiki/World-Café) oder GridWorkshop.

Letzteres ist ein Workshop, an den für die einzelnen Punkte - vergleichbar zum Worldcafe – einzelne Flipcharts erstellt werden, die dann gleichzeitig (!) von den Teilnehmern timeboxed (!) bearbeitet werden. Es werden mehrere Zyklen je 15-20 Minuten inszeniert, die Teilnehmer schreiben ihre Überlegungen dazu auf Post-its und kleben sie zu den jeweiligen Themen-Flipcharts (folgend gehen wir von einem GridWorkshop aus).

Ein typische Ablauf eines solchen Grid-Workshops könnte wie folgt sein:

- Vorstellung (Teilnehmer)
- Workshop Vorgehen
- Runde 1 & 2
- Pause
- Runde 3 & 4
- Diskussion einzelne Themen in der Gruppe (mit Live Korrektur + Ergänzungen)
- Next steps

Erfahrungsgemäß braucht man für die Vorstellungsrunde und Workshop Vorgehen rd. 20 Minuten. Runden 1 – 4 plus Pause (je 15 min.) macht in Summe 75 min. Für die Diskussion und Zusammenfassung der Flipchart würde ich im Schnitt je 5 min ansetzen (macht im konkreten Beispiel 60 min.). Next Steps und Zeit-Puffer rd. 30 min --> dh. in Summe kann ein GridWorkshop gut und gern in 3 bis 3,5 Stunden gemacht werden.

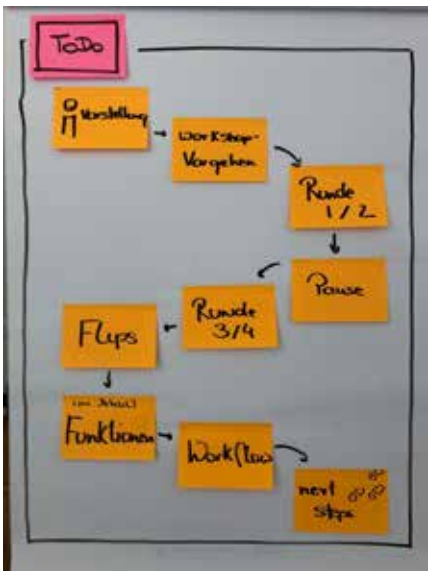
Empfehlung Nr. 2: Egal welchen Workshop Sie machen, gehen Sie aus „Veranstalter" zumindest zu zweit hin. Teilen Sie die Aufgaben in Moderation und Dokumentation.

Kommen wir nun zu den einzelnen Themenbereichen, die Sie vorbereiten und im Rahmen des Workshops abarbeiten sollten:

Am Start einer Prozess- oder Organisationsunterstützung durch ein Tool stellt sich schnell die Frage nach dem **richtigen Tool**. Diese Thematik hat Alexander Vukovic bereits in seinem Vortrag zur Open Source Testautomationstool-Auswahl (SEQIS Blog: https://www.seqis.com/de/events/10things_open-source-testautomationstools-in-der-projektpraxis?day=20100429&times=1272492000,1272578399 hervorragend beschrieben, deshalb gibt es in diesem Blog keine weiterführenden Kommentare.

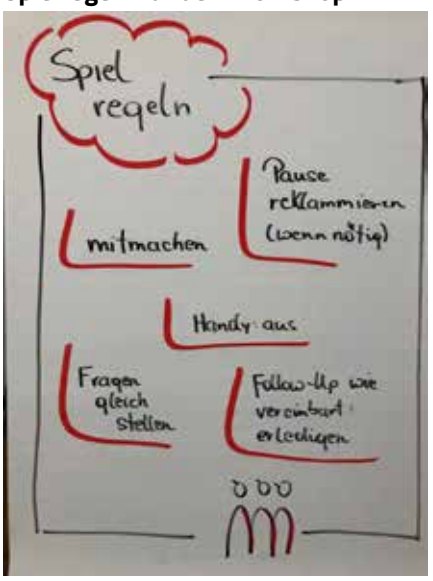
Das Format **GridWorkshop** ist eine SEQIS Eigenentwicklung. Im Kern geht es darum, einen Raum mit unterschiedlichen Themen-inseln/-ankern zu schaffen, die typische Frontal-präsentation zu vermeiden und die Teilenehmer dazu zu bringen, sich zu „bewegen" – ***im Raum***, durch die verstreuten Themeninseln, **umfassend**, dh. jeder für jedes Thema um den Knowhow-Abgleich abzusichern, ***inhaltlich***, weil im Gesamtbild der Punkte gut erkennbar ist, wo ggf. verhärtete Positionen („ich will aber") Alternativen brauchen. ***Timeboxed*** stellt sicher, dass der Zeitbedarf in einen rahmen bleibt, der für alle leicht machbar ist.

Die Agenda



Für strukturierte Teilnehmer ein Muss. Es hat sich bewährt, die Agenda als PostIts zu gestalten und dann im Laufe des Workshops – wenn einzelne Punkte „done" sind – auch visuell zu „erledigen".

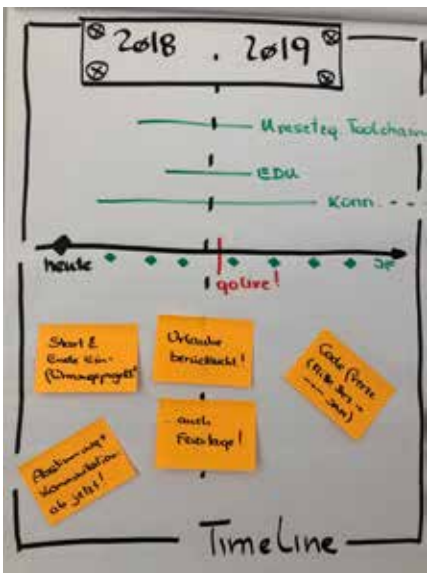
Spielregeln für den Workshop



... na klar, Spielregeln sind ein „Muss". Bitte diese jedenfalls in der Vorstellungsrunde des Workshops vorstellen und Zustimmung durch die Teilnehmer abholen. Falls aus dem Ruder läuft – strenger Blick auf die Spielregeln, kurz innehalten... Und zumeist geht's dann schon wieder besser weiter.

Verlassen wir nun den „Rahmen" und kommen wir zu den inhaltlichen Punkten und Empfehlungen...

Timeline



Skizzieren Sie die Timeline (heute, zyklische Projektmeetings & Treffen, Milestones und Go Live) und tragen Sie bereits bekannte Phasen (Zeitspanne der Realisierung, erste Zyklen in Produktion, Kommunikation, Ausbildung, usw.) ein. Stellen Sie sicher, dass auch dieses Flip mit PostIt-Gedanken ergänzt und kritisiert wird. Stellen Sie z.B. die Fragen nach kritischen Phasen im Projekt, nach Störungen („Weihnachtsgeschäft", „Jahresabschluß", usw.)!

Ziele

... werden leider zu oft nur mit „kennen wir ja eh" abgetan. Dahinter sind aber handfeste Kriterien für Erfolg/Misserfolg und zumeist haben die einzelnen Stakeholder-Vertreter sehr unterschiedliche Wahrnehmungen, was die Ziele sind.
Geben Sie also diesem Thema entsprechend Raum, instrumentieren Sie bereits von Ihnen als bekannt Eingestuftes und schreiben (!) Sie diese Punkte mit einer noch offenen Checkbox auf das Flip. Im Rahmen der Flip-Vorstellung sollten Sie dann diese Punkte auch noch einzeln besprechen und bei Zustimmung in der Checkbox abhaken oder durchstreichen.

Empfehlung Nr. 3: Achten Sie bei der Vorstellung der Ziele auf Rückmeldungen – Gemurmel, Körperhaltung und ähnliches lassen uU auf Zielkonflikte schließen! Sie sollten diesen in diesem Rahmen zumindest festhalten (wenn sie sich nicht gleich lösen lassen, „agree to disagree") und in Folge abarbeiten.

Prinzipien

Prinzipien, also die Grundsätze, für die Entwicklung einer Toolchain können sehr unterschiedlich sein. Es sind oftmals nicht funktionale Anforderungen, die nach Fertigstellung im Sinne einer Erfolgsprüfung nochmals untersucht werden soll; folgende Prinzipien habe ich in meiner Praxis sehr oft an dieser Stelle besprochen:

- „kann mit angemessenen Zeitaufwand gepflegt werden"
- Sprache deutsch
- Integrierte Onlinehilfe
- Ausbaufähig, erweiterbar
- Selbsterklärend (= sehr geringer Schulungsbedarf)
- So gut wie Möglich im Standard bleiben

Knowhow Transfer

Wie soll die Einführung und der Knowhow Transfer für Neulinge der Toolchain gemacht werden? Stimmen Sie an Schulungsvarianten „Train-the-Trainer" vs „Schulungsorganisation" bzw. Keyuser-Trainings ab. Zertifizierungen bringen auch sehr oft eine fundierte Ausbildung „im Package" mit – falls das möglich ist. Darüber hinaus bieten Online-Hilfe, Erklärungsvideos (mit wesentlichen Punkte aus der Anwendungspraxis) und Hotline auch gute Möglichkeiten, die Anwender auf das „Neue" vorzubereiten bzw. sie auch laufend zu unterstützen.

Vergessen Sie jedoch bitte nicht, dass Sie ggf. neben dem Training der Anwender auch noch Ihre System-

betreuer schulen und trainieren müssen. Sichern Sie sich ggf. auch externe Unterstützung, insbesondere wenn die Toolchain von Externen entwickelt wurde.

Und auch für das „Umfeld" sollten Sie neben dem Knowhow Transfer auch eine Information a la „Was ändert sich durch den Einsatz von XY?" erstellen und z.B. im Intranet platzieren.

Analyse der „Player"

Kennen Ihre Stakeholder die Plattform, auf der Sie Ihre Lösung aufbauen? Ja? Nein? Diese Antwort hat eine unmittelbare Auswirkung auf z.B. den Schulungsaufwand bei der Einführung. Aber auch andere Parameter sollten erhoben und beurteilt werden, starten Sie mit der Erhebung der wesentlichen Stakeholder-Gruppe und gehen Sie dann ins Detail:

- Wie ist die Einstellung von XY (z.B. Buchhaltung) zum vorliegenden Projekt?
- Welche Bedeutung hat die Toolchain für diese spezifische Stakeholder-Gruppe?
- Wie war die Zusagen-Verlässlichkeit dieser Gruppe für welche Arbeiten in der Vergangenheit? Müssen wir ggf. Erinnerungsfunktionen oder vergleichbares einrichten?

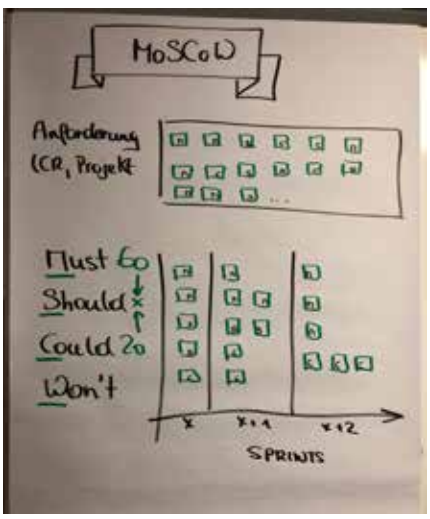
Empfehlung Nr. 4: Bei dieser Analyse werden ggf. Kritiken an den „anderen" geäußert. Achten Sie auf eine möglichst neutrale Formulierung dieser Einschätzung, damit Ressentiments gleich von Anfang an verhindert werden!

Toolchain @ Ihrem Unternehmen
Nehmen wir an, bei der konkreten Realisierung setzen Sie auf Atlassian Jira und Confluence. Sind diese Systeme bereits in Ihrem Unternehmen eingesetzt? Ja? Für diesen Fall sollten Sie folgende Punkte bei dem Workshop abstimmen:

- Wie kommen wir zu Space bzw. Project ?
- Welche Version – PlugIns – Lizenzen liegen vor?
- Ist für die externe Unterstützung ein VPN-Zugriff einzurichten
- Welche Umgebungen gibt es? TEST, PROD...?
- Mit welchen Personen/ Administratoren ist zu sprechen? Nun, idealerweise sind diese Über den Workshop informiert und machen sogar mit... :-)

Wenn Sie andere Systeme oder Lösungen einsetzen: Oben stehende Liste gibt Ihnen eine gute Ausgangsbasis für die relevanten Fragen zu Ihrer Lösung.

MoSCoW



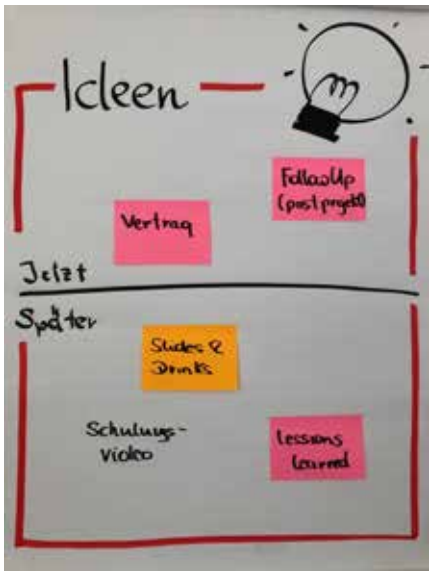
Kaum ein seriöses Projekt kommt heute ohne einer fundierten Reihung von ToDo's durch. Führen Sie eine Priorisierung nach MUST – SHOULD – COULD WONT (= MoSCoW) durch und erklären Sie diesen Priorisierungsansatz im Rahmen des Workshops.

Toolchain Projekt
Welche spezifischen Fragestellung sollten Sie im Rahmen Ihres Toolchain-Realisierungsprojekts abstimmen?

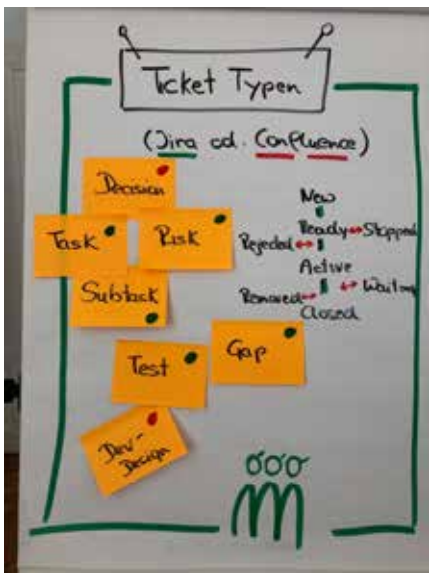
Für das Projekt sind Antworten – neben den unzähligen, die ganz allgemein von den Projekt Management-Frameworks wie PRINCE 2 oder AgilePM empfohlen werden – zu folgenden Fragen erfahrungsgemäß besonders relevant:

- Welche Reaktionszeiten brauchen wir im Rahmen der unterschiedlichen Phasen (Analyse, Implementierung, Test, Migration, Go Live, Post Go Live)?
- Wie soll eine kontinuierliche Abstimmung stattfinden? Daily's, JourFixes mit Status-präsentationen, Management Reports,...?
- Wie wollen und sollen wir Projekt-Status kommunizieren?
- Stichwort Externe: Kann die Toolchain vor Ort realisiert werden – oder besser remote? Was immer definiert wird: Beides bedingt Infrastruktur, also eben Räume oder technische Connectivity...

Ideen für Jetzt oder später
Bereits in diesem ersten Workshop werden „Ideen" kommen, mit denen Sie nicht gerechnet haben oder die nachfolgend erst zugeteilt werden können. Versuchen Sie zumindest den Inhalt der Idee zu verstehen und diese in die Tasche für „Jetzt" oder aber auch „Später" zuzuteilen.



Ticket-Typen
Jede Toolchain besteht aus eine Art Ticket (z.B. Issuetypes im Jira). Versuchen Sie eine erste Sammlung unterschiedlicher Tickets zusammenzustellen. Ergänzen Sie diese um spezifische Informationen (Attribute), die mit diesem Tickettyp einhergehen. Darüber hinaus sollten Sie auch erste grobe Workflows / Statusketten dieser Tickettypen abzustimmen und zu visualisieren.



Funktionen
Clustern Sie Ihre Toolchain in unterschiedliche funktionale und nicht funktionale Blöcke und sammeln Sie funktionale Anforderungen bzw. Anmerkungen und Kommentare. Diese Aufstellung dient der Visualisierung der unterschiedlichsten Anforderungen – damit sichern Sie ein gemeinsames Bild zur Aufgabenstellung ab. Aber Achtung: Erfahrungsgemäß wird diese Übersicht keine abschließende Erfassung sein sondern nur ein Startpunkt und etwas mehr Detail zur Aufgabenstellung!

Der richtige Startpunkt für eine „Definition of Done“
Auch wenn Sie diesen Punkt – genauso wie die „Funktionen“ - nicht in diesem Workshop spontan vollständig schaffen werden: Viele Teilnehmer kommen bereits mit klaren (Teil)Zielvorstellungen. Holen Sie sich diese ab. ■

Mag. (FH) Alexander Weichselberger hat seine Einsatzschwerpunkte in den Bereichen Systemanalyse, Software Test, Koordination und Management von exponierten Großprojekten und kann auf jahrelange Erfahrung zurückblicken.

Dieses Wissen gibt er gerne in Form von Coachings, Methodentrainings und Fachvorträgen weiter.

Alle Termine im Überblick:

10.04.2019
ATB Expertentreff
Tech Gate, Wien

06.06.2019
SEQIS „10 things“
Expertentreff:
„Business Analyse“,
Tech Gate, Wien

26.06.2019
ATB Expertentreff
Tech Gate, Wien

Die Anmeldung ist ab sofort möglich!

Weitere Infos:
www.SEQIS.com

April		
1	Mo	
2	Di	
3	Mi	
4	Do	
5	Fr	
6	Sa	
7	So	
8	Mo	
9	Di	
10	**Mi**	**SEQIS @ ATB Treff**
11	Do	
12	Fr	
13	Sa	
14	So	
15	Mo	
16	Di	
17	Mi	
18	Do	
19	Fr	
20	Sa	Karfreitag
21	So	
22	Mo	Ostermontag
23	Di	
24	Mi	
25	Do	
26	Fr	
27	Sa	
28	So	
29	Mo	
30	Di	

Haben Sie schon Ihre nächste Weiterbildung geplant?

www.SEQIS.com

Mai		
1	Mi	Staatsfeiertag
2	Do	
3	Fr	
4	Sa	
5	So	
6	Mo	
7	Di	
8	Mi	
9	Do	
10	Fr	
11	Sa	
12	So	
13	Mo	
14	Di	
15	Mi	
16	Do	
17	Fr	
18	Sa	
19	So	
20	Mo	
21	Di	
22	Mi	
23	Do	
24	Fr	
25	Sa	
26	So	
27	Mo	
28	Di	
29	Mi	
30	Do	Christi Himmelfahrt
31	Fr	

Juni		
1	Sa	
2	So	
3	Mo	
4	Di	
5	Mi	
6	**Do**	**SEQIS „10 things“**
7	Fr	
8	Sa	
9	So	Pfingstsonntag
10	Mo	Pfingstmontag
11	Di	
12	Mi	
13	Do	
14	Fr	
15	Sa	
16	So	
17	Mo	
18	Di	
19	Mi	
20	Do	Fronleichnam
21	Fr	
22	Sa	
23	So	
24	Mo	
25	Di	
26	**Mi**	**SEQIS @ ATB Treff**
27	Do	
28	Fr	
29	Sa	
30	So	

Über die SEQIS Expertentreffs „10 things I wished they'd told me!“

An Informationen mangelt es meist nicht – im Gegenteil, derer gibt es oft mehr als genug. Wichtiger denn je ist es, an die entscheidenden Informationen zu gelangen. Als Dienstleistungsunternehmen sind wir uns unserer Rolle als Informant bewusst und sprechen die an Software Test und IT Analyse Interessierten mit der Veranstaltungsreihe „10 things I wished they'd told me!“ konkret an.

Für all jene, die Software entwickeln, nutzen, beschaffen, in einem Betrieb für die Softwarequalitätssicherung zuständig oder als Requirements Engineer/Business Analyst tätig sind, haben wir eine passende Plattform geschaffen!

Bei unseren Expertentreffs erhalten Sie die Möglichkeit branchenbezogene Erfahrungen auszutauschen und wertvolle Tipps von den Profis abzustauben. Die Vortragenden bringen aktuelle Test- und IT Analyse-Themen auf jeweils 10 knackige Punkte und teilen mit Ihnen ihre Erfahrungen aus zahlreichen großen und komplexen IT Projekten.

Save-the-Date zu den „10 things“ 2019

Auch im Jahr 2019 laden wir Sie wieder ein, unsere vier kostenlosen Expertentreffs zu aktuellen IT Trendthemen zu besuchen.

Agile Transformation: 10 Erfolgsfaktoren aus der Praxis
Donnerstag, 14. März 2019

Done

Business Analyse: Kick-Down gleich von Start weg
Donnerstag, 06. Juni 2019

Wie anpassungsfähig ist Ihre IT?
Ein Einstieg in die Resilienz
Donnerstag, 19. September 2019

Testen in Software Lifecycle Virtualization -
die Zukunft des Testings?
Donnerstag, 14. November 2019

Melden Sie sich gleich an und sichern Sie sich Ihren Platz:
www.SEQIS.com/de/events-index

SEQIS „10 things" Expertentreff #3/2018: Automate your mobile

von *Christina Eder*

Bei den ersten „10 things" nach der Sommerpause 2018 gab Senior Consultant Markus Schwabeneder 10 instruktive Tipps zur Testautomation mobiler Endgeräte.

Für die Softwareentwicklung sind Apps für Smartphone und Tablet aus dem Arbeitsalltag nicht mehr wegzudenken. Moderne Systeme müssen mittlerweile unbedingt auch über mobile Geräte bequem verfügbar sein.

„Die automatisierte Qualitätssicherung dieser Apps steckt noch immer in den Kinderschuhen. Hier wird viel Potential verschenkt.", so Markus Schwabeneder. „Testautomation für mobile Geräte ermöglicht beispielsweise Fast Feedback, also schnelle Regressionstests, wesentlich weniger manuellen Aufwand bei Regressionstests und die Möglichkeit, Tests auf vielen verschiedenen Devices durchzuführen." Grundsätzlich gelten die gleichen Regeln wie beim „normalen" Testen – nur mit komplexeren Randbedingungen. Um das Potenzial bestmöglich zu nutzen, hat er seine „10 things" zusammengestellt:

1. Bauen Sie eine Pyramide!

Achten Sie auf gute Unit-Testabdeckung und weniger Oberflächenautomation.
Auch bei mobilen Tests gilt die „ideal software testing pyramid"

2. Fahren Sie auf den Hauptstraßen!

Der typische Gebrauch der App wird intensiv getestet.
Tests, die die gesamte Applikation inklusive der UI prüfen, können Probleme oder Fehler aufdecken, die von Tests kleinerer Einheiten nicht entdeckt werden. Darum sind End-to-End-Tests notwendig.
End-to-End Tests sind aber deutlich schwieriger zu warten, aufwändiger zu implementieren und meistens auch deutlich langsamer in der Durchführung. Darum sollte man sie auf ein vernünftiges Maß reduzieren.
Die Funktionen, die besonders häufig von den Usern benutzt werden, oder die ganz besonders wichtig sind, werden „Hauptstraßen" genannt.
Eine gute Faustregel ist es, nur diese Hauptstraßen End-to-End zu testen.

3. Riskieren Sie richtig!

Schätzen Sie die Eintrittswahrscheinlichkeit und den Schaden bei Eintritt ab.
Zur Bestimmung des Testaufwands hilft es, zu bestimmen, wie kritisch sich ein spezifischer Fehler auswirken würde und wie wahrscheinlich so ein Fehler sein könnte.
Die Teile, für die sich hier ein großes Risiko ergibt, müssen besonders intensiv getestet werden.
Umgekehrt sind exzessive Tests von Anforderungen, die kein großes Risiko bergen, nicht sinnvoll.

4. Kennen Sie Ihre Kunden!

Testen Sie die richtigen Geräte und in der richtigen Umgebung.
Um die Hauptstraßen zu bestimmen und die Auswirkungen von eventuellen Problemen in der App abschätzen zu können, müssen Sie wissen, wie die App von ihren Anwendern benutzt wird.
Ebenfalls ist es wichtig zu wissen, auf welchen Geräten und unter welchen Betriebssystemen die App eingesetzt wird.
Das Anwenderverhalten und besonders die Geräte-Betriebssystem-Kombination können sich mit der Zeit ändern. Darum ist es sinnvoll, zu analysieren, auf welchen Geräten welche Funktionen benutzt werden.

5. Kombinieren Sie clever!

Planen Sie ausreichende Abdeckung ein und wägen Sie

Aufwand- und Kosteneffizienz ab.
In dem Dschungel der mobilen Endgeräte ist es unmöglich, alle denkbaren Kombinationen zu testen. Es gibt aber diverse Möglichkeiten, eine große Testabdeckung mit relativ wenigen, aber gut gewählten Tests zu erreichen.
Eine gute und noch relativ einfache Möglichkeit dazu nennt sie „Pairwise Testing" (auch: „Pairwise-Methode" oder „Paarbildungs-methode").
Ähnlich dem „Pairwise Testing" gibt es noch eine Reihe komplizierter Verfahren, die auch gewisse Vorteile bieten können. Diese Verfahren können unter dem Begriff „Orthogonal Array Testing" zusammengefasst werden.

6. Nutzen Sie die Cloud!

So haben Sie den Vorteil großer Geräteauswahl bei überschaubaren Kosten.
Das Testen auf eigenen physischen Geräten bedeutet, dass man sich mit dem Geräteankauf und auch der Gerätewartung auseinandersetzen muss.
Ein Weg, diesen Problemen aus dem Weg zu gehen, ist, sich die Geräte quasi zu „mieten".
Große Player in der IT (Amazon (AWS Cloud), Google (Firebase Test Lab) und Microsoft (Xamarin Test Cloud)) bieten solche Services an.
Es gibt auch Firmen, die genau darauf spezialisiert sind (u.a. Experitest, Kobiton, Perfecto, SauceLabs).

7. Verbessern Sie die Tests kontinuierlich!

So können Sie vom sofortigen Nutzen profizieren und Probleme in der Testautomation schnell erkennen.
Vermeiden Sie die lange Entwicklung einer „Big Bang"-Lösung, sondern starten Sie mit kleinen Schritten, die aber sofort Nutzen bringen!
Starten Sie mit den Tests Ihrer kritischen Anforderungen!
Leben Sie das „Pfadfinder Prinzip": Verlassen Sie jeden Ort schöner als Sie ihn betreten haben. Das bedeutet, jeder neue Code soll nicht nur Funktionalität hinzufügen, sondern auch den alten Code lesbarer, wartbarer und testbarer machen. Auch die Einführung von Unit-Tests kann auf diese Weise gut vorangetrieben werden.

8. Testen Sie in Stufen!

So können Sie schnelle Tests in der CI und umfassende Tests vor dem Go-Live umsetzen.
Es ist sinnvoll, unterschiedliche Teststufen zu definieren. So können z.B. Unit-Tests direkt nach einem Check-In vom Code laufen.
Für alle automatisierte Tests, die eine längere Laufzeit haben, bieten sich „Nightly Builds" an.
Tests, die bei jedem Durchlauf Kosten verursachen, können in Stufen erfolgen, die nur manuell ausgelöst werden.

9. Starten Sie die Qualitätssicherung schon in der Entwicklung!

Moderne Standards führen zu leichter testbarem Code.
Folgende Prinzipienverringern das Fehlerrisiko und erhöhen die Testbarkeit des Codes:

- MVC/MVVM/MVP
- DRY
- SOLID
- Single responsibility principle
- Open/Close principle
- Liskov substitution principle
- Interface segregation principle
- Dependency inversion principle
- Verwendung von Automation IDs

10. Halten Sie die Testautomation aktuell!

Mustern Sie nicht mehr benötigte Testfälle aus und bewerten Sie Geräte- und/oder Betriebssystemabdeckung regelmäßig.
Mobile Testautomation erfordert ständige Wartung:

- Anpassung aufgrund neuer Anforderungen an die App
- Anpassungen aufgrund verändertem Userverhalten
- Anpassung aufgrund von Neuentwicklungen
- Wartung der physischen Geräte
- Anpassung auf neue Betriebssysteme
- Softwareupdates
- Ständige Verbesserungen
- U.v.m. ■

Auf www.seqis.com/youtube finden Sie das Interview zur Veranstaltung

MMag. Christina Eder ist Marketing Managerin bei SEQIS. Sie ist erste Ansprechpartnerin für Presse- und Marketinginformationen.

Von Drucksortengestaltung über Videodreh und -schnitt, klassischer Pressearbeit, Betreuung der SEQIS Onlinekanäle bis hin zur Organisation von Veranstaltungen übernimmt sie Marketing- und Kommunikationsagenden.

Besonders wichtig ist ihr die stetige Entwicklung von Medien, Tools und Konzepten.

Appium - Die Lösung für mobile Testautomation?

von *Andreas Steiner*

Mit dem Angebot unterschiedlicher Betriebssysteme und Endgeräte, vor allem im mobilen Anwendungsbereich, gewinnen plattformübergreifende Applikationen, kurz App, immer mehr an Bedeutung.

Kunden erwarten, wenn eine App auf Android verfügbar ist, diese auch auf iOS verwenden zu können und umgekehrt. Darum wird immer öfter die Entscheidung getroffen Hybrid-Apps zu entwickeln, welche für Apples iOS-Geräte, genauso wie auf Googles Androiden laufen. Für die Entwicklung solcher cross-platform Apps gibt es mittlerweile eine handvoll Tools, wie beispielsweise:

- Xamarin
- Phonegap
- Sencha

Aber welches Tool verwendet man, um seinen Multiplattformer zu testen?

Für automatisierte Tests stellen Apple, genauso wie Google Automations-Frameworks für ihre Produkte zur Verfügung.

- Ab iOS 9.3: XCUITest
- Ab Android 4.2: UIAutomation 2

Android-Tests sind ausschließlich in Java zu schreiben, für iOS-Tests kann man zwischen Objective-C oder Swift wählen. Da Appium mit Libraries der populärsten Programmiersprachen versorgt ist, können Tests in folgenden Sprachen erstellt werden:

- Ruby
- Python
- Java
- JavaScript
- PHP
- C#

Worldwide, Dec 2018 compared to a year ago:

Rank	Change	Language	Share	Trend
1	↑	Python	25.36 %	+5.2 %
2	↓	Java	21.56 %	-1.1 %
3	↑	Javascript	8.4 %	+0.0 %
4	↑	C#	7.63 %	-0.4 %
5	↓↓	PHP	7.31 %	-1.3 %
6		C/C++	6.4 %	-0.4 %
7		R	4.01 %	-0.3 %
8		Objective-C	3.21 %	-0.9 %
9		Swift	2.69 %	-0.7 %
10		Matlab	2.06 %	-0.3 %
11	↑↑	TypeScript	1.65 %	+0.2 %
12	↓	Ruby	1.57 %	-0.3 %

- Objective C
- Robotframework

Appium baut auf die Web-Driver API und erweitert diese um sämtliche von den Herstellern angebotenen Frameworks, sowie nützliche Smartphone-Funktionen (Zoomen, Scrollen etc.). Dies bringt den Vorteil, dass die App nicht instrumentiert (Anm: dh. nicht mit Zusatzinformationen erweitert und damit verändert) werden muss. Es wird also die App getestet, die letztendlich auch im Store landet.

Durch das verwendete Client-Server-Protokoll (JSON Wire Protocol) für den WebDriver, wird ausschließlich über HTTP-Requests

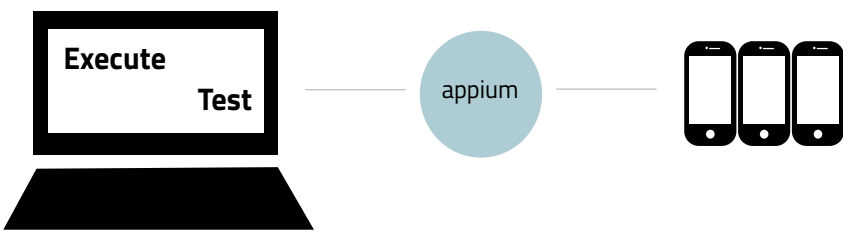


Abbildung 2: Auf einem Client wird ein Test-Run gestartet. Appium übersetzt die Befehle und führt diese auf den entsprechenden Smartphones durch. Nach Abschluss des Testlaufs, werden die Ergebnisse an den Client übermittelt.

kommuniziert. Es können Tests in jeder Programmiersprache geschrieben werden, solange sie eine Client-HTTP-API unterstützt. Allerdings ist es ratsam die von Appium zur Verfügung gestellten Libraries zu verwenden.
Die größte Stärke der verwendeten Client-Server Architektur ist, dass Tests und Server auf unterschiedlichen Geräten laufen können. Mithilfe von Cloud Services werden Tests auf einer Vielzahl echter aber auch simulierter Geräte durchgeführt. So können sich Kosten der Testumgebung senken lassen, ohne einen Qualitätsverlust befürchten zu müssen. Ganz im Gegenteil, es kann sogar bedeuten, dass sich dadurch die Qualität der App beträchtlich erhöht, vor allem, wenn zuvor nur auf den aktuellsten Flagships oder nur mit Emulatoren getestet wurde.

Mittlerweile werden auch Client-Lösungen für Appium angeboten. Beispielsweise bietet Appium Desktop eine einfache Oberfläche, einen Inspektor und wie aus anderen Tools bekannt, kann per Maus-Klick ein Testfall aufgenommen werden. Diese Client-Lösungen schränken einen in der Anwendung zwar oft ein, zeigen aber wie vielfältig Appium eingesetzt werden kann.
Außerdem: Appium ist Open Source und hat eine große Community hinter sich vereint. Es gibt selten ein Problem, welches gänzlich unbekannt ist oder für das es noch keine Lösungsvorschläge gibt. Es ist vertreten auf GitHub und seit 2016 Teil der JS Foundation. ■

Alle Infos zu Appium gibt es auf https://appium.io

4 philosophische Grundsätze, auf denen Appium aufbaut:

1. **You shouldn't have to recompile your app or modify it in any way in order to automate it.**
2. **You shouldn't be locked into a specific language or framework to write and run your tests.**
3. **A mobile automation framework shouldn't reinvent the wheel when it comes to automation APIs.**
4. **A mobile automation framework should be open source, in spirit and practice as well as in name!**

Können Sie sich mit diesen Grundsätzen identifizieren?

Andreas Steiner ist Consultant für Software Test bei SEQIS.

Die Schwerpunkte seiner Projekterfahrung liegen in den Bereichen Testautomation, Testdurchführung, Softwarequalitätssicherung und Requirements Engineering.

Kommunikation, Teamwork sowie vorausschauendes Arbeiten sind für ihn essenzielle Bestandteile für einen reibungslosen Projektablauf.

Toolchain V 1.0

von *Hansjörg Münster*

Wenn wir von Toolchain reden, meinen wir in der Regel die vielen Helferlein, die unsere Arbeit erleichtern. In meinem Fall als Testmanager sind die digitalen Tools heute aus unserem Alltag nicht mehr wegzudenken: Ticketsysteme steuern unseren Arbeitsalltag, Testtools speichern unsere Testfälle und führen sie manchmal automatisch aus, die Requirements werden in Applikationen verwaltet und diese füttern gleich das Backlog. Viele tolle Dinge stehen uns heute zur Verfügung, als Beispiel soll hier nur die großen Touch-Displays erwähnt sein, vor denen wir während des Standup's andächtig stehen und virtuelle Kärtchen per Fingertipp verschieben.

Aber vergessen wir nicht auf die kleinen Helferlein des Alltags - althergebracht, analog und selbstverständlich.

Der erste Weg in der Früh nach dem Einschalten des PC's führt mich zur Kaffeemaschine - absolut notwendig für einen Kaffeejunkie wie mich. Espresso doppelt - zumindest für mich. Kollegen haben es lieber etwas milder, vielleicht, wenn es eine dieser ganz modernen Maschine ist mit automatischem Milchschaum. Inzwischen sind diese Automaten so zum Standard geworden, sodass Filterkaffee (zum Glück) schon die Ausnahme ist.

Danach mache ich meine Tagesplanung - nicht mit Jira, Wunderlist, Googles ToDos - nein ganz klassisch mit einem Post-it und Kugelschreiber. Post-its sind aus meinem Alltag sowieso nicht weg zu denken.

Der Schreibtisch ist voll beklebt, hinter mir an der Mauer ein haptisches Taskboard und wenn ich einen Kollegen an seinem Platz nicht antreffe, hinterlasse ich gerne ein gelbes Selbstklebe-Zettelchen. Nur: Es müssen die superklebenden Spezial Post-its („super sticky") sein, damit man sie öfter auf andere Stellen kleben kann und nicht ein Windstoß aus dem haptischen Taskboard einen bunten Zettelhaufen am Boden macht. Habe ich schon die Notwendigkeit einer Kaffeemaschine erwähnt? Ein zu Ende gehender Vorrat an Kaffee, Zucker oder Milch etc., kann mit einem PostIt ganz einfach auf dem Kaffeeautomaten für die meist frühmorgendlichen HelferInnen der Office-Betreuung mitgeteilt werden.

Eine ähnliche Aufgabe erfüllt das Notizbuch. Im Gegensatz zu den Post-its hat das Geschriebene im Notizbuch aber einen dauerhaften Charakter. Viele verwenden dafür teure gebundene Bücher, manche in Leder, mit Aufklebern zum Beschriften, einem Schnürchen als Lesezeichen. Mir sind aber die ganz einfachen Collegeblöcke lieber. In einem Kasten in meinem Keller sammle ich die vollgeschrieben Blöcke seit vielen Jahren und obwohl ich noch nie in einem älteren Block was gesucht oder benötigt habe, kann ich mich nicht davon trennen. Vielleicht geben diese Blöcke ein schönes Lagerfeuer bei meiner Pensionierungsfeier?

Damit die Tools Post-it und Notizbuch funktionieren, benötigen wir noch Stifte. Die Auswahl von Stiften, Bleistiften und Kugelschreiber ist für mich ein ewiges Thema: Oft gehe ich in ein Schreibwarengeschäft und probiere unterschiedliche - meist höher preisige - Schreibgeräte aus. Und leider kaufe ich diese dann oft auch. Wobei die Wahl des Schreib-

Quelle: Adobe Stock; Urheber: Andrey Popov

gerätes durchaus gewissen Mode-erscheinungen unterliegt: Vor einigen Jahren waren klassische Füllfedern in Gebrauch, im Moment verwende wieder ich einen einfachen Kugelschreiber. Notwendig und immer in meiner Tasche sind auch ein Bleistift und ein Textmarker. Und damit ich den Marker auch auf den gelben PostIt verwenden kann, habe ich einen neon-orangen Highlighter.

Damit sind meine persönlichen Tools 1.0 schon alle erwähnt. Aber es gibt natürlich weitere, die jedoch allgemein oft zur Verfügung stehen. Da wäre mal das klassische Whiteboard. Zu einem guten Whiteboard gehören auch Whiteboardstifte - und nur Whiteboardstifte. Oft liegen auf der Ablage Stifte in allen Farben, Dicke und Größen, aber es sind oft auch Permanent-Stifte darunter. Ärgerlich, wenn man nach einem intensiven Brainstorming oder Workshop merkt, dass man sich im wahrsten Sinn auf der Tafel ver-"ewigt" hat. Lästig ist es auch, wenn die Stifte undicht sind oder kleckern. Nicht nur ein Hemd hat bei mir das zeitliche gesegnet, weil der Stift tropfte.

Ähnlich zu einem Whiteboard sind auch Flipcharts. Sobald ich mit jemandem ein fachliches Detail diskutiere, zieht es mich oft zum Flip um zu zeichnen, skizzieren, visualisieren. Ärgerlich, wenn dann alle Blätter bereits benutzt sind. Und oft hilft dann die übliche Schnelllösung nicht, den Block umzudrehen, weil diese Idee schon jemand vorher hatte. Wäre doch nett, wenn man den Nachfolgenden ein sofort benutzbares Flipchart hinterlassen würde. Und außerdem ist das auch ein nicht zu unterschätzendes Sicherheitsthema: Gebrauchte Flips hängen lassen - da freuen sich die Nachfolgende über die Lektüre (und Wirtschaftsspione oder DSGVO-Jäger soll es ja auch geben).

Habe ich schon erwähnt, dass das wichtigste Tool, die Kaffeemaschine ist? Und wenn diese in der Nähe der Besprechungsräume ist, wäre es perfekt! ■

Titel: Eichhörnchen, Künstler: Michael König, Technik: Buntstift-Zeichnung

Hansjörg Münster ist Principal Consultant und Teamlead bei SEQIS.

Als Allrounder deckt er ein breites Spektrum an Aufgaben ab. Die Schwerpunkte seiner Tätigkeit liegen in den Bereichen Test Management, Testautomation und Lasttest.

Ganz oben auf der Prioritätenliste des IT Profis steht, einen Nutzen und Mehrwert in der Qualitätssicherung seiner IT Projekte zu generieren.

Toolchain in DevOps

von *Karin Wagner*

DevOps (aus Development, Entwicklung und IT-Operations, IT-Betrieb) ist eine Philosophie, welche die Zusammenarbeit und Kommunikation der Bereiche Entwicklung und Betrieb fördern soll. Die Umsetzung des DevOps-Ansatzes wird häufig empfohlen, wenn Continuous Delivery eingeführt werden soll. Toolchain (Werkzeugkette) bezeichnet eine Reihe von Programmen die – meist in der Softwareentwicklung – genutzt werden, um ein Produkt zu entwickeln und auszuliefern.

Die beiden Begriffe DevOps und Toolchain werden mittlerweile häufig in einem Atemzug genannt. Wie genau hängen sie zusammen? Gibt es „die Toolchain für DevOps"? Worauf sollte man bei der Tool-Auswahl achten?

DevOps

Um eine reibungslose Zusammenarbeit zwischen Development und Operations zu erreichen, definiert John Willis, ein Veteran der DevOps-Bewegung, fünf Grundprinzipien der Management-Strategie:

- *Culture:* Schaffen von gegenseitigem Vertrauen; stetiger Informationsfluss; Lernbereitschaft
- *Automation:* Automatisierung bestimmter Arbeitsvorgänge, um die Anwendungsentwicklung zu beschleunigen
- *Lean:* Vermeiden von Verschwendung; Wertgenerierung; Transparenz; ganzheitliche Prozessoptimierung
- *Measurement:* Festlegung einheitlicher Bewertungskriterien (auch über die Applikation und deren Komponenten hinaus)
- *Sharing:* Bereitschaft der Mitarbeiterinnen und Mitarbeiter, ihr Wissen zu teilen, voneinander zu lernen und Erkenntnisse proaktiv mitzuteilen.

Aus diesen Prinzipien geht deutlich hervor, dass für die erfolgreiche Umsetzung von DevOps zwei Komponenten besonders essentiell sind: einerseits die Haltung und Einstellung aller involvierten Personen und andererseits die Automatisierung von Arbeitsvorgängen.

Toolchain

Der Begriff „Toolchain" bedeutet wörtlich Werkzeugkette. Dies impliziert, dass die einzelnen Tools wie die Glieder einer Kette aneinandergereiht sind und nacheinander zum Einsatz kommen. Mit den richtigen Tools können Prozesse sowohl gestartet als auch abgeschlossen werden.

Entscheidet sich ein Unternehmen für Automatisierung, soll manuelle Routinearbeit eliminiert werden. Teams profitieren dadurch von reproduzierbaren Prozessen und erstellen zuverlässige Systeme.

Bevor jedoch Prozesse automatisiert werden, ist ein Blick auf die aktuelle Toollandschaft ratsam. Häufig haben Unternehmen keine einheitlichen Systeme im Einsatz. Sowohl in der Entwicklungsabteilung als auch im Bereich IT Betrieb wird eine Reihe von jeweils unterschiedlichen Tools in eigenen Umgebungen verwendet.

Im Rahmen des Aufbaus einer Toolchain ist es daher essentiell, die Toollandschaft zu analysieren und zu harmonisieren. Dabei ist es wichtig, die unterschiedlichen Anforderungen der Teams an das jeweilige Tool zu kennen. In den meisten Fällen erfordert die Einführung von DevOps die Integration neuer Tools.

Toolchain für DevOps

Die Entscheidung, welche Tools auch in Zukunft verwendet werden sollen bzw. welche abgelöst werden sollen, hängt von mehreren Aspekten ab:

- Gute Kommunikation und Zusammenarbeit sind grundlegend für den Erfolg von DevOps-Teams, daher ist bei der Auswahl der Tools darauf zu achten, dass diese beiden Faktoren gefördert werden.
- Neben der Erfüllung der Kommunikationsanforderungen ist sicher zu stellen, dass die Werkzeuge über Planungsfunktionen, Analyse-Tools und Funktionen zur Testautomation sowie zur Bereitstellung von Testdaten verfügen.
- Kommunikation ist nicht nur im Hinblick auf Tools, sondern vor allem zwischen den Beteiligten wichtig. Die Unternehmenskultur – das Miteinander, die Haltung der einzelnen Personen – trägt maßgeblich zum Erfolg von DevOps bei. Daher sind alle Beteiligten in den Entscheidungsprozess einzubeziehen.
- Um die Tools gemäß der vorgesehenen Verwendung nutzen zu können, müssen die Mitarbeiterinnen und Mitarbeiter entsprechend rechtzeitig in der Bedienung der Tools geschult werden.

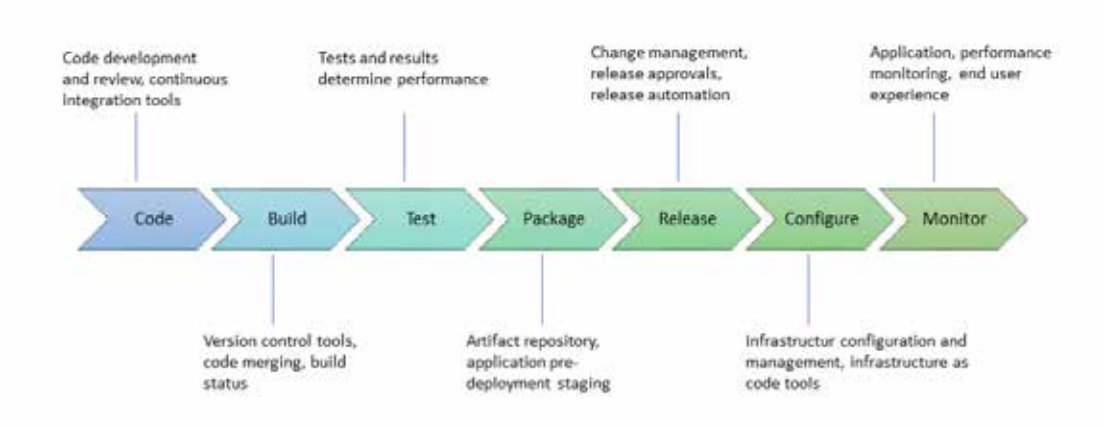


Abbildung 1 DevOps lifecycle stages (Quelle: http://www.pontine.ch/devops/devops-toolchain-review/)

Neben der Frage, welche Programme im Unternehmen genutzt werden sollen, muss außerdem geklärt werden, welche der Tools nur in einem Bereich zum Einsatz kommen werden und welche sowohl von Dev als auch von Ops verwendet werden sollen. Sebastian Müller, Softwareentwickler, schlägt in seinem Fachartikel für die sigs-datacom.de vor, folgende Tools für beide Bereiche vorzusehen:

- Monitoring
- Tool zum Zugriff auf Logfiles
- Tool für Datenbankauswertungen
- Automatische Ausführung von DDL-Scripten (Data Definition Language)
- Tool zum Lesen und (wenn als sinnvoll erachtet) Schreiben von Firewall-/Routing-Regeln
- Web-Analytics
- SEO-Tools (search engine optimization)
- Tools zum automatischen Initiieren von Applikationen

Fazit

Um DevOps erfolgreich einzuführen, erleichtern die richtigen Tools die Erreichung der Unternehmensziele. „Richtig" können Tools allerdings immer nur für das jeweilige Unternehmen oder Vorhaben und Projekt sein. Es gibt sie nicht, „die Tool-Chain für DevOps".
Die Unternehmenskultur ist im Rahmen der Einführung von DevOps ein essentieller Faktor. Software-Tools allein sind nicht in der Lage, für die erforderliche Agilität, Kommunikation und Zusammenarbeit zu sorgen.

Mindestens genauso wichtig wie die Tools ist es daher, die Mitarbeiter und Mitarbeiterinnen in möglichst viele Entscheidungsprozesse – auch und besonders in die Auswahl der unterstützenden Tools, denn die sorgen am Ende für die tägliche Zufriedenheit.■

Quellen:
https://t3n.de/news/was-bedeutet-eigentlich-devops-723440/
https://de.atlassian.com/devops
https://www.sigs-datacom.de/uploads/tx_dmjournals/mueller_OS_DevOps_14_japT.pdf
http://www.pontine.ch/devops/devops-toolchain-review/

Karin Wagner war nach ihrer Ausbildung dem Schwerpunkt Software Engineering viele Jahre als Consultant im Bereich ERP-Systeme tätig.

Bei SEQIS ist sie Expertin für IT Analyse und Software Test. Sie verschafft sich in Projekten rasch Überblick und ist dadurch eine kompetente Ansprechperson für die einzelnen Bereiche.

Sie arbeitet gerne in Teams und mag Herausforderungen, die analytisches Denken erfordern.

SEQIS „10 things“ Expertentreff #1/2019: Agile Transformation

von *Christina Eder*

Bei den 10 things im März 2019 verriet Alexander Vukovic seine persönlichen top 10 Tipps und Tricks, wie eine agile Transformation gelingen kann.

Auf Basis der Erfahrungen mit unzähligen agilen Transformationsprojekten in den letzten rund 20 Jahren präsentierte Alexander Vukovic die 10 wichtigsten Erfolgsfaktoren, die sich in der Praxis bewährt haben.

„Eine agile Transformation ist mehr als eine agile Transition. Es geht nicht darum, etwas organisatorisch umzustellen, sondern eine intrinsisch motivierte Änderung der Einstellung und des Verhaltens der Mitarbeiter und des Umfelds herbeizuführen.“, so Alexander Vukovic, SEQIS Founder und agile Experte.

Im Rahmen des Expertentreffs wurden beispielsweise Fragen nach dem Legacy Code und dem Umgang mit Knowhow Kopfmonopolen beantwortet, der richtige Umgang mit Fehlern beleuchtet und viel über das richtige Mindset der Mitarbeiter sowie den hohen Stellenwert von intrinsischer Motivation diskutiert. Die 10 Tipps zusammengefasst:

Tipp 1: Transformation durch intrinsische Motivation

Analysieren Sie Ihre Situation und versuchen Sie durch intrinsische Motivation eine Agile Transformation anstatt einer Agile Transition durchzuführen. Damit ändern Sie das Mindset der Mitarbeiter nachhaltig.

Tipp 2: Transformationsstrategie für Legacy Code

Berücksichtigen Sie die Herausforderung richtig mit bestehenden Legacy Code in Ihrer Transformationsstrategie umzugehen. Schaffen Sie kurz- und langfristige Lösungen, wie mit dieser Erbschaft umgegangen werden soll.

Tipp 3: Know-how Monopole abbauen

Lösen Sie Know-how Monopole durch Trainingsmaßnahmen, dem Buddy-Konzept und einer Community of Practice auf.

Tipp 4: Fehler nutzen statt verurteilen

Schaffen Sie Rahmenbedingungen, die das Erkennen, Eingestehen und Lernen aus Fehlern fördern, z.B. durch Team Retrospektiven und „Feedback Burger“.

Tipp 5: Stakeholderinteressen und Anforderungen priorisieren

Die Priorisierung der Interessen der Stakeholder und deren Anforderungen darf nicht am Entwickler abgeladen werden. Schaffen Sie eine geeignete Rolle, wie z.B. Product Manager, Product Owner oder Business Visionary/Advisor, die diese Abstimmungen übernimmt.

Tipp 6: Die richtigen Rahmenbedingungen vorgeben

Setzen Sie Rahmenbedingungen, die dem Team die Richtung vorgeben und es dem Team erlauben, Verantwortung für ihre Arbeit zu übernehmen.

Tipp 7: Agilen Prozess gemeinsam mit dem Team passend zum Kontext definieren

Definieren Sie den agilen Prozess gemeinsam mit Ihrem Team und

einem agilen Coach passend zu Ihrem Kontext; Scrumban mit MoSCow ist eine Möglichkeit.

Tipp 8: Nutzen Sie ein physisches Board

Integrieren Sie ein physisches Board in Ihren Prozess, denn Wissensarbeit ist persönlich und sollte visualisiert kommuniziert werden.

Tipp 9: Kommunizieren Sie kontinuierlich und deutlich. Schaffen Sie Ehrlichkeit und Transparenz im Projekt!

Entlasten Sie das Team durch systematische Automation der repetitiven und fehleranfälligen Tätigkeiten: Unit Test, Build, Deployment und Monitoring. Automatisierung der Entwicklung beinhaltet folgende Schritte:

- Continuous Testing
- Continuous Integration
- Continuous Deployment
- Continuous Delivery & Monitoring

Tipp 10: Definition of Done für die Transformation

Erkennen Sie, wann die Transformation abgeschlossen ist und reduzieren Sie die Einflussnahme von außen. Scheuen Sie sich nicht davor, einen zweiten oder dritten Versuch für eine Transition zu starten, wenn der erste Anlauf scheitert.

Folgende Aspekte können für eine Definition of Done beispielhaft angenommen werden:

- Weniger Rückfragen
- Man wird selbst nicht mehr gebraucht
- Teams entwickeln sich von selbst weiter
- Der Prozess wird gelebt
- Die Transformation ist nachhaltig ■

MMag. Christina Eder ist Marketing Managerin bei SEQIS. Sie ist erste Ansprechpartnerin für Presse- und Marketinginformationen.

Von Drucksortengestaltung über Video-dreh und -schnitt, klassischer Pressearbeit, Betreuung der SEQIS Onlinekanäle bis hin zur Organisation von Veranstaltungen übernimmt sie Marketing- und Kommunikationsagenden.

Besonders wichtig ist ihr die stetige Entwicklung von Medien, Tools und Konzepten.

Toolchain unchained: Agile Product Management mit Aha!

von *Alexander Vukovic*

Jira ist überall. Mit Jira hat Atlassian ein Werkzeug geschaffen, das leichtgewichtig aber dennoch mächtig ist. Über Plugins lässt sich das einfache Ticketingsystem zu einem Testmanagement oder Supportportal aufrüsten.

Auch im Bereich des Produktmanagements gibt es diverse Plugins, die Jira um Planungsfunktionen und besseres Requirement Engineering erweitern.

Eines davon ist Aha! https://www.aha.io laut Eigen definition die #1 bei den Roadmappingsystemen weltweit. Wir haben Aha! mit einigen anderen eigenständigen Systemen und Jira Plugins verglichen und letztlich als die beste Lösung für den Kundenkontext gemeinsam mit unserem Kunden ausgewählt.

Wie sich im Zuge des Projektes gezeigt hat, ist Aha! Eigentlich kein Plugin, sondern ein eigenständiges System, das sehr stark mit Jira integriert. Dafür ist kein Plugin notwendig, sondern die Jira Webhooks müssen dafür aktiviert werden. So werden Änderungen in beiden Systemen bidirektional synchronisiert.

Aber was hat Aha! nun was anderen Plugins oder einem nackten Jira fehlt?

Aha! bildet den gesamten Produktmanagementprozess für den Product Owner/Produktmanager ab.

Produktstrategie

Zuerst definiert man in Aha die Produktstrategie. Dafür bietet Aha! diverse Werkzeuge, beginnend mit der Definition der Produktstruktur,

JIRA

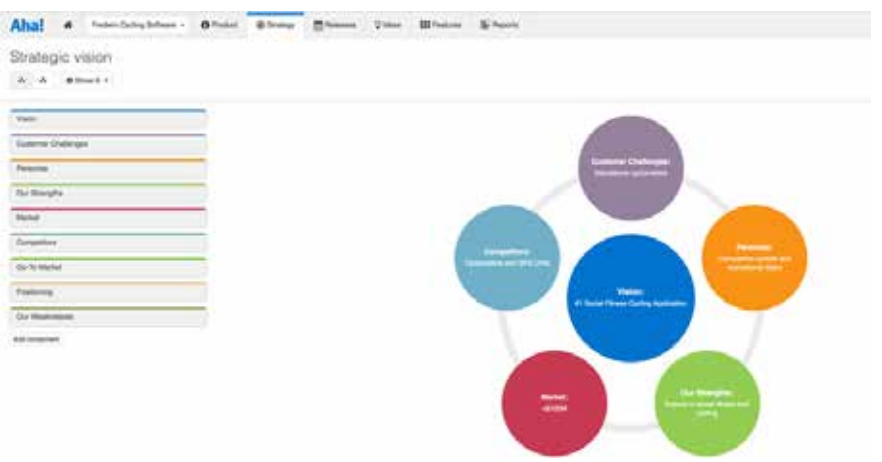
Jira ist eine Webanwendung zur Fehlerverwaltung, Problembehandlung und operativem Projektmanagement. Jira wird auch in nicht-technischen Bereichen für das Aufgabenmanagement eingesetzt. Sie wurde von Atlassian entwickelt. Primär wird Jira für die Softwareentwicklung eingesetzt.

Quelle: Wikipedia

die baumartig aufgebaut wird. Für jedes Produkt wird dann eine Vision erstellt. Die Vision besteht aus jeweils einem Satz für:

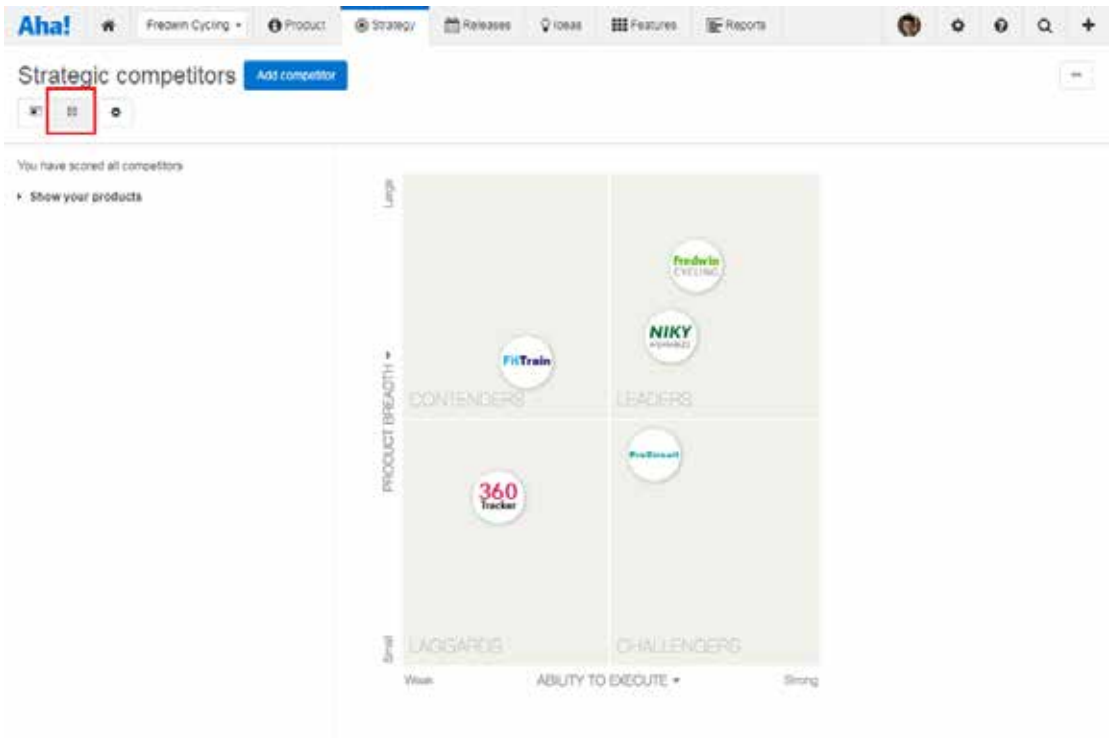
- Vision
- Mission
- Position
- Stärken
- Schwächen
- Chancen
- ...

Die Elemente können konfiguriert und angepasst werden.



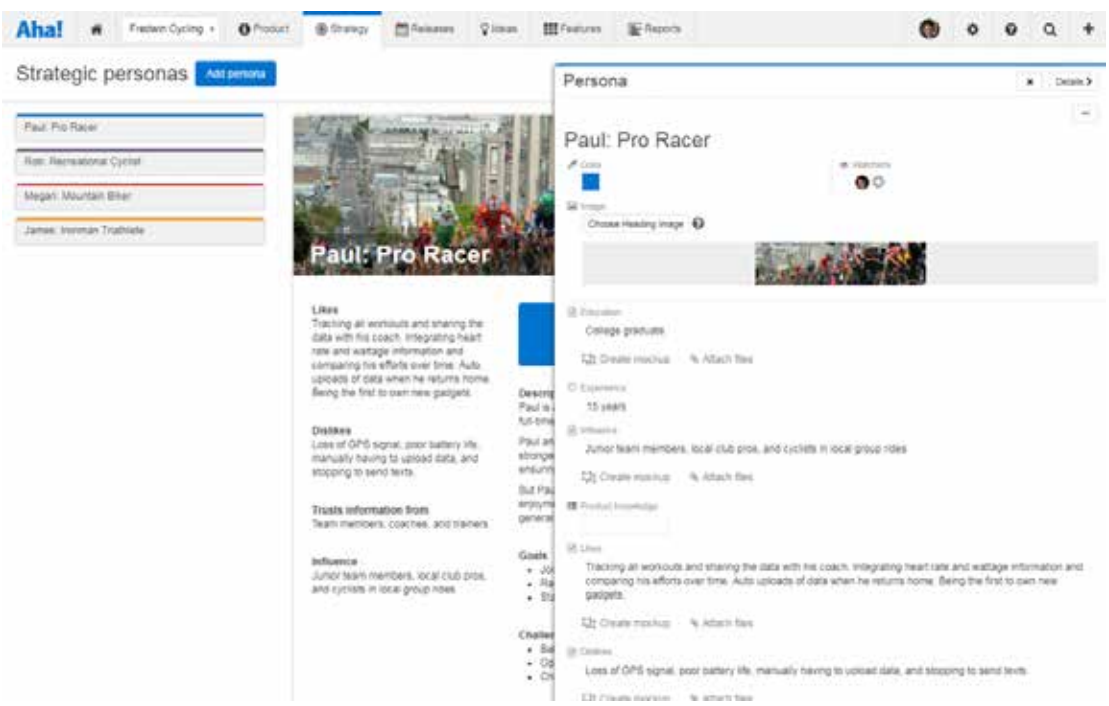
Aha! Beispiel für eine strategische Produktvision.

In einem eigenen Bereich für die Mitbewerber können deren Stärken und Schwächen erfasst und in Vergleichsmatrizen gegenübergestellt werden.



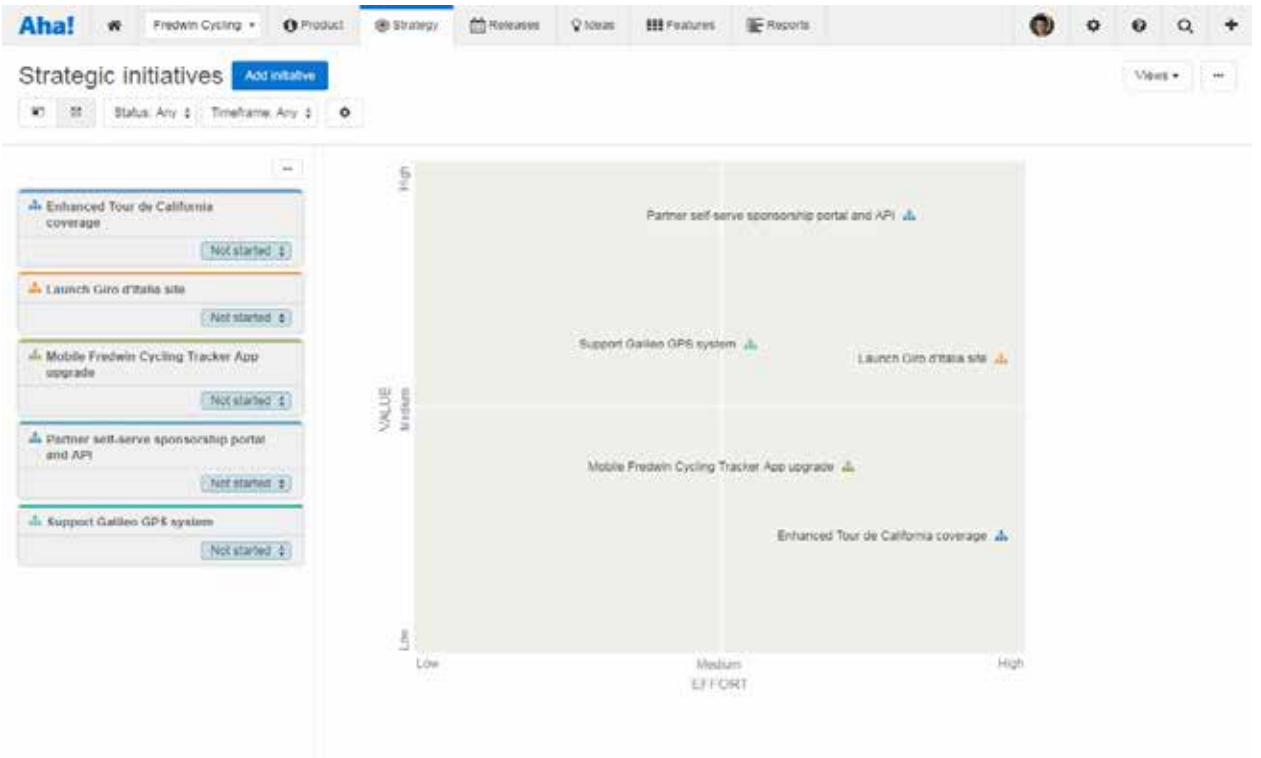
Aha! Erfassung und Stärken/Schwächen Analyse der Mitbewerber

Ebenso erfasst Aha! Personas, also künstliche erschaffene Personen, die eine Personengruppe repräsentieren.

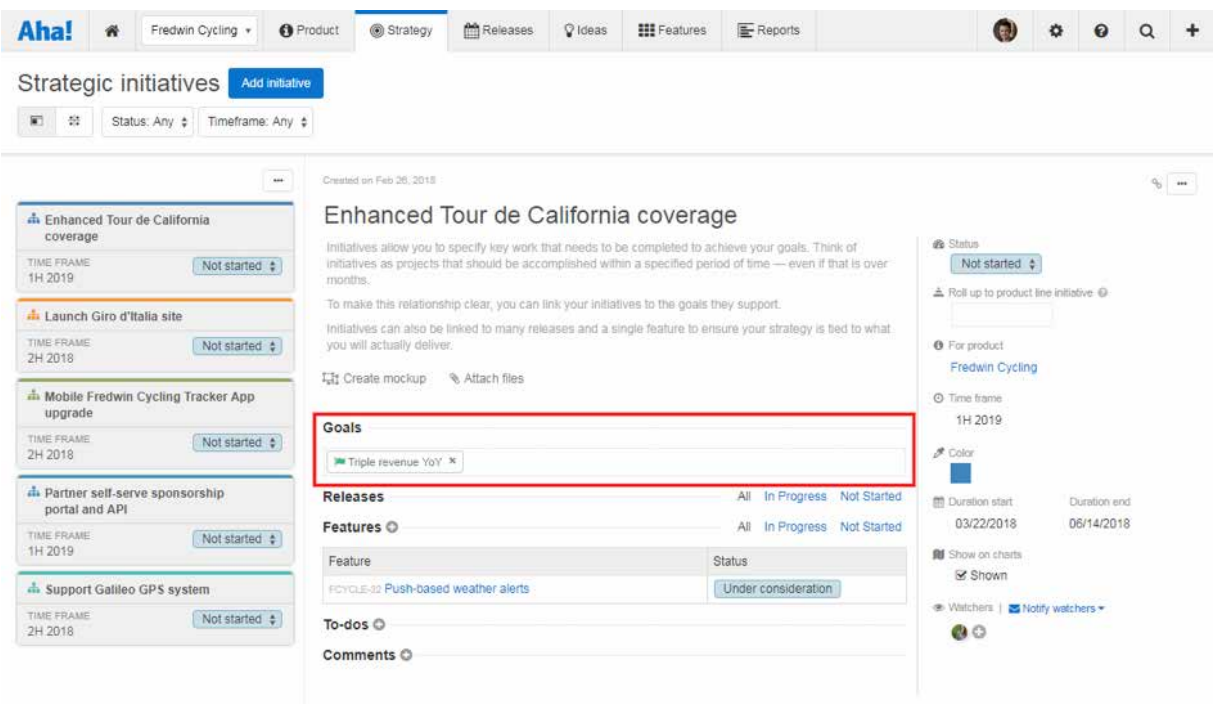


Aha! Beispiel für die Persona Paul: Pro Racer

Die Produktstrategie wird durch strategische Initiativen und Ziele komplettiert. Diese Initiativen und Ziele dienen auch als Kategorisierung und Zuordnung der zukünftig zu beschreibenden Anforderungen an das System.



Aha! Beispiel für eine strategische Produktvision.



Aha! Erfassung der strategischen Produktinitiativen und der konkreten Ziele dahinter.

Aha! Automatisch aktualisiertes Burndown Chart

Das Erfassen der Produktstrategie ist eine unabdingbare Voraussetzung, um im Anforderungsdschungel Richtung und Orientierung zu haben. Auf die Product Owner wirkt eine nicht schaffbare Menge an Änderungswünschen, Verbesserungsvorschlägen und Bugreports ein. Sie müssen daher priorisieren und abwägen können, welche Dinge dennoch Platz bekommen. Die Produktstrategie unterstützt dabei.

Releaseplanung

Systematische Produktentwicklung für viele Kunden (im konkreten Fall ca. 1500 aktive Kunden), bedingt verlässliche und berechenbare Auslieferungszyklen an alle Kunden.

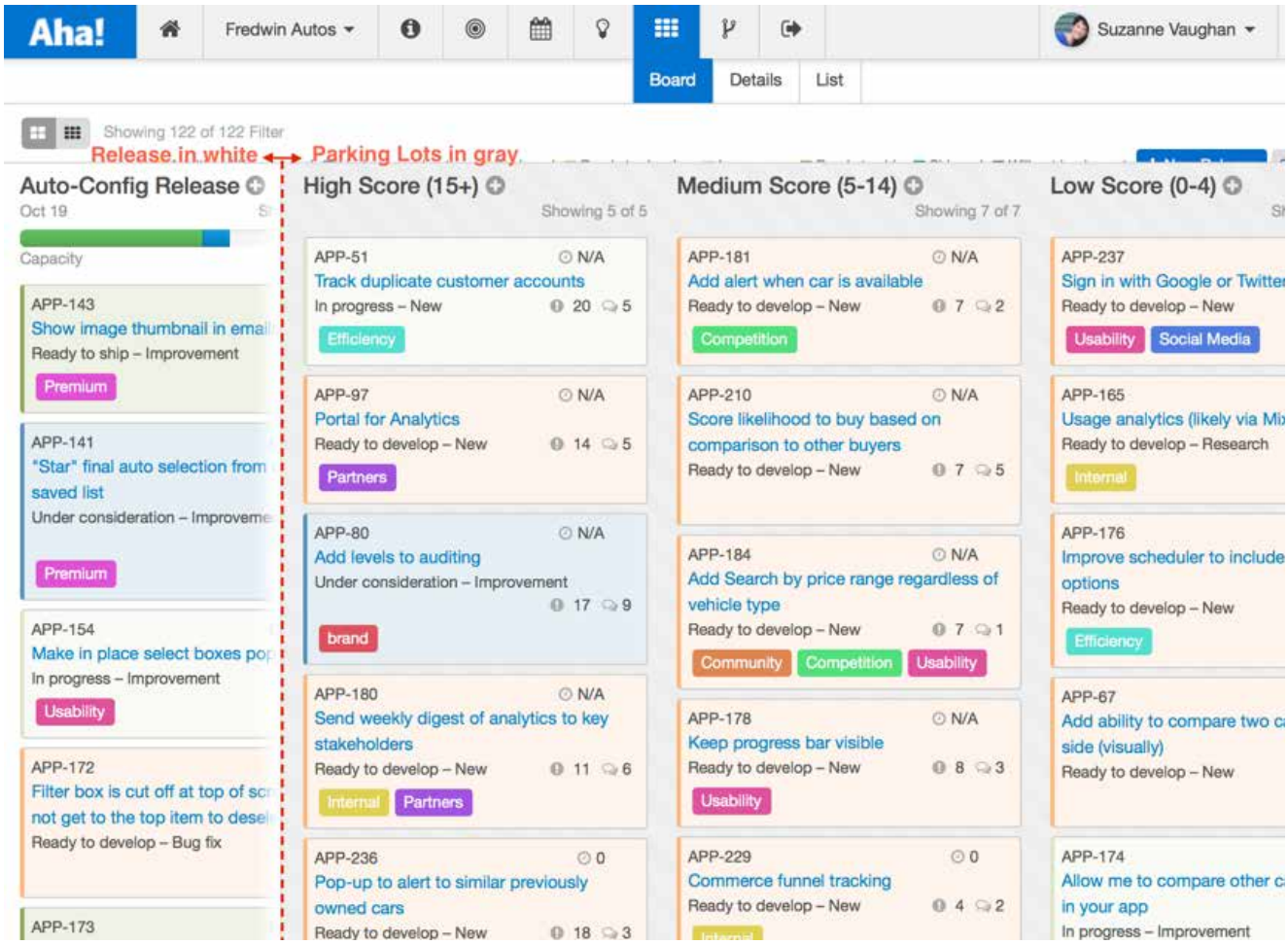
Dabei unterstützt Aha! mit der integrierten Releaseplanung. Releases können auf unterschiedlichen Granularitätsebenen geplant werden. Masterrelases fassen die große Menge an Änderungen nochmals zusammen und erlauben es später automatisiert eine jederzeit aktuelle Roadmap für alle Kunden bereitzustellen.

Releases können in Meilensteine und Phasen unterteilt werden und unterliegen einem eigenen Releaseworkflow. Die Planung kann in unterschiedlichen Sichten bis hin zu einer Retro-Gantt-Ansicht dargestellt werden. Der Release-fortschritt wird aggregiert grafisch dargestellt, kann aber auch im Detail als Burndownchart visualisiert werden.

Mit sogenannten Parking Lots werden alle Zustände außerhalb konkreter Releases bezeichnet. Hier kann ein Backlog definiert und befüllt werden, oder über Selektion unterschiedliche Sichten auf die ungeplanten Features hergestellt werden. Das erleichtert massiv die Planung der Releases.

Der Screenshot (unten) zeigt das Featureboard, auf dem die gesamte Planung mittels Drag & Drop erfolgt. Dabei kann auch die Teamkapazität in abstrakten Storypoints oder Zeit eingeplant werden.

Aha! arbeitet unabhängig von der gewählten agilen Methode nicht z.B. mit Stories und Epics, sondern mit Features und Master Features. Diese können über den Featuretyp nochmals genauer spezifiziert werden. Die Karten und die darauf angezeigten Daten können umfangreich konfiguriert werden, um dem Product Owner eine fundierte Planung zu ermöglichen.

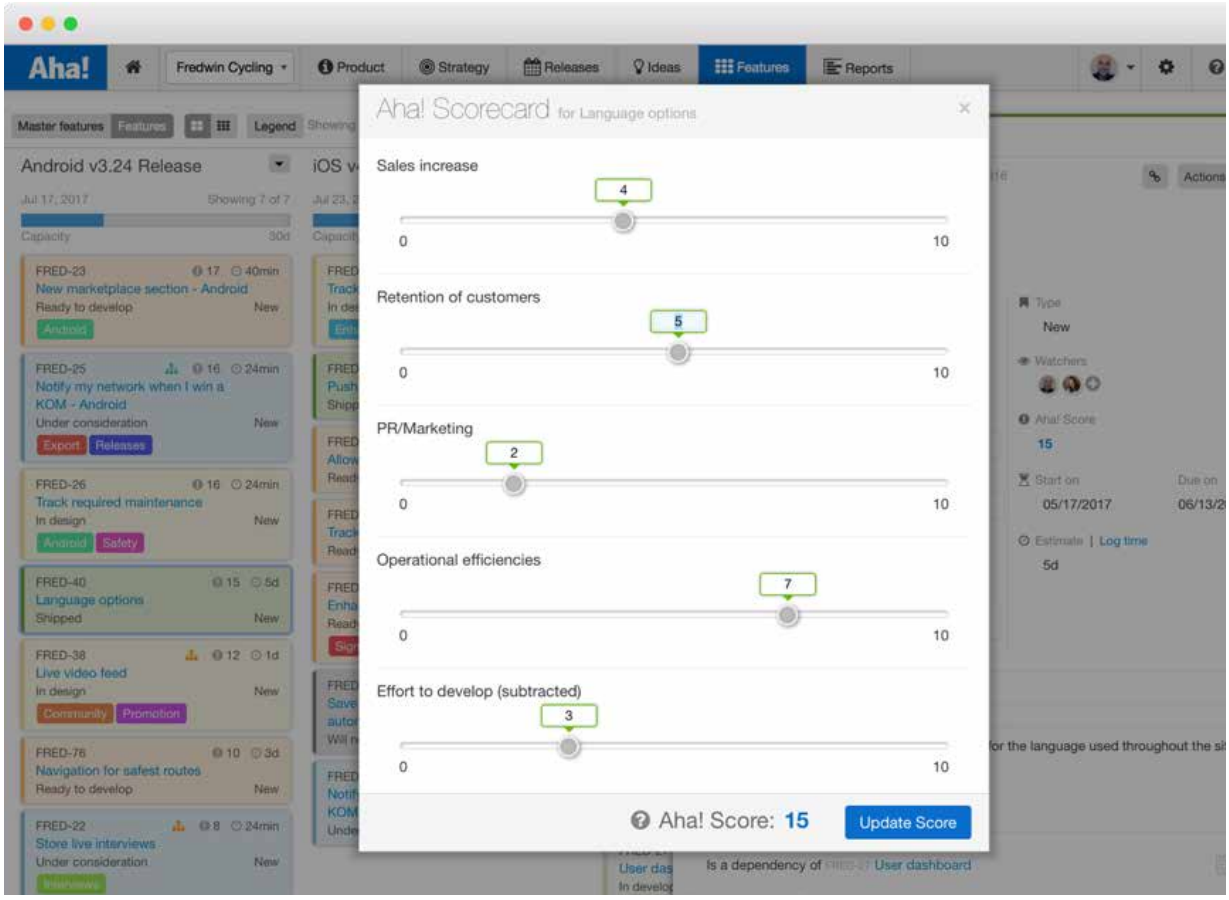


Aha! Parking Lots zeigen die ungeplanten Features am Featureboard automatisch gruppiert nach Feature Score

Features und Master Features

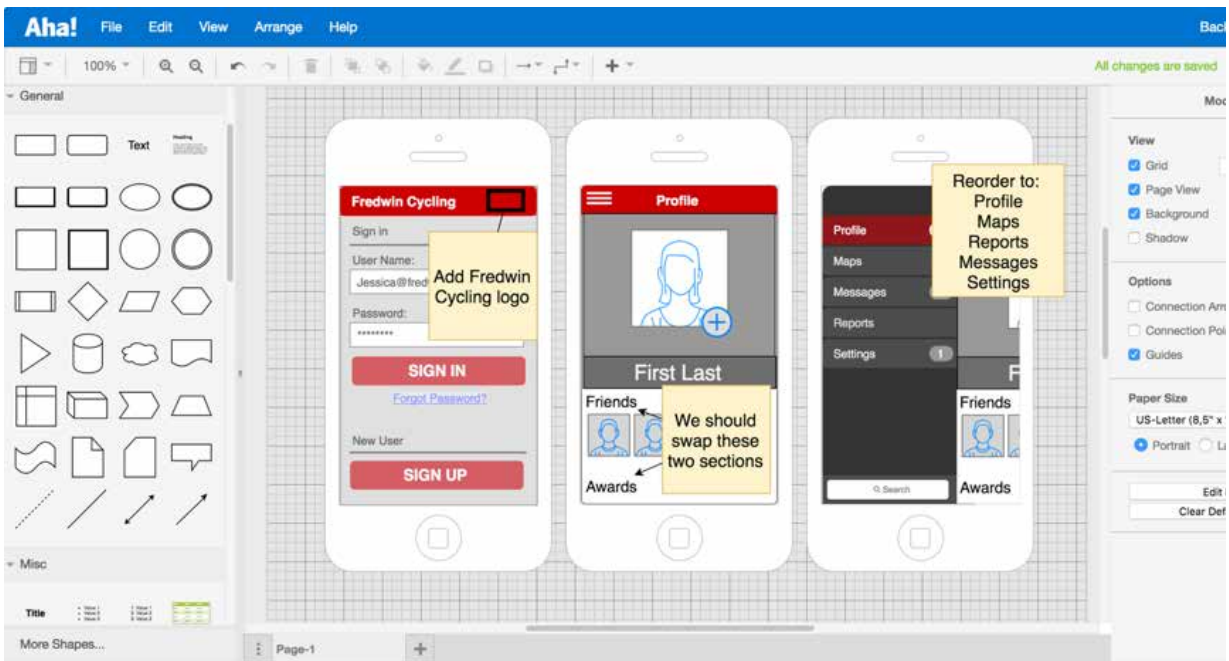
Features und Master Features unterliegen ähnlich wie in Jira einem konfigurierbaren Workflow (so wie alle Elemente auch in Aha!). Es gibt Tags, Kommentare, Likes, einen Assignee und eine Ticket-ID. All das ähnelt sehr stark Jira und erleichtert letztlich auch die Integration in Jira.

Eine Funktion, die das Produktmanagement stark vereinfacht ist die Aha! Scorecard. Sie ermöglicht Kriterien und Wertigkeiten zu definieren und so die Berechnung einer Business Value zu vereinfachen. Man kann eigene Scorecards und Berechnungen konfigurieren und so wirklich den Kundenkontext abbilden.



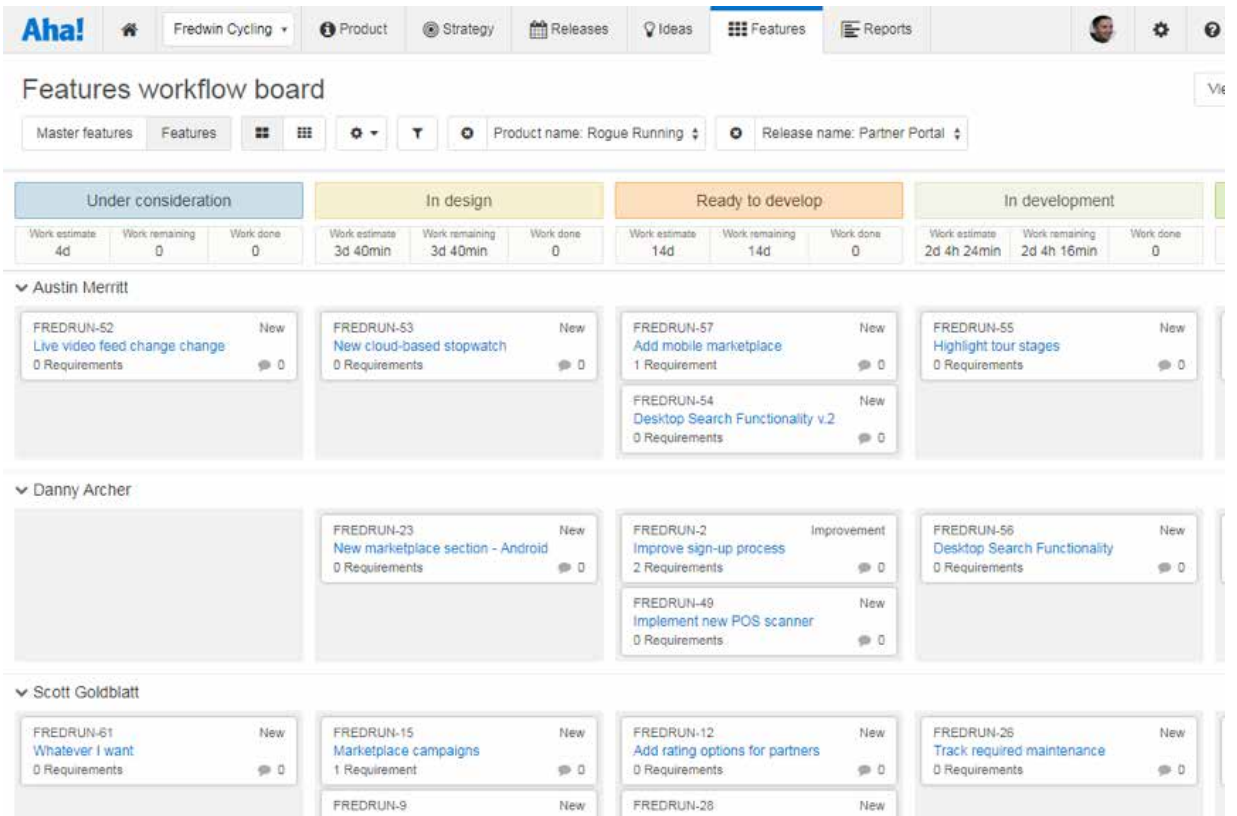
Aha! Beispiel für eine Scorecard, die den Aha! Feature Score berechnet.

Sehr häufig werden Anforderungen heutzutage an Mockups definiert und verfeinert. Aha! bietet direkt integriert ein Mockupwerkzeug, das die Definition direkt bei jedem Feature erlaubt.



Aha! integrierte Mockups

Sehr häufig werden Anforderungen heutzutage an Mockups definiert und verfeinert. Aha! bietet direkt integriert ein Mockupwerkzeug, das die Definition direkt bei jedem Feature erlaubt.



Aha! Features auf dem Kanbanboard

Jira Integration

Die Jiraintegration von Aha! ist äußerst mächtig. Dabei können Entitäten auf der Seite von Aha! auf Issuetypes auf Jiraseite bis auf Feldebene + Feldwerte gemappt werden.

Ist die Synchronisation erst einmal eingerichtet, können die Anforderungen und das Backlog-management komplett in Aha! durchgeführt werden. Die Entwicklung, Testing und Abnahme erfolgt in Jira und der Status ist dennoch auch jederzeit in beiden Systemen aktuell.

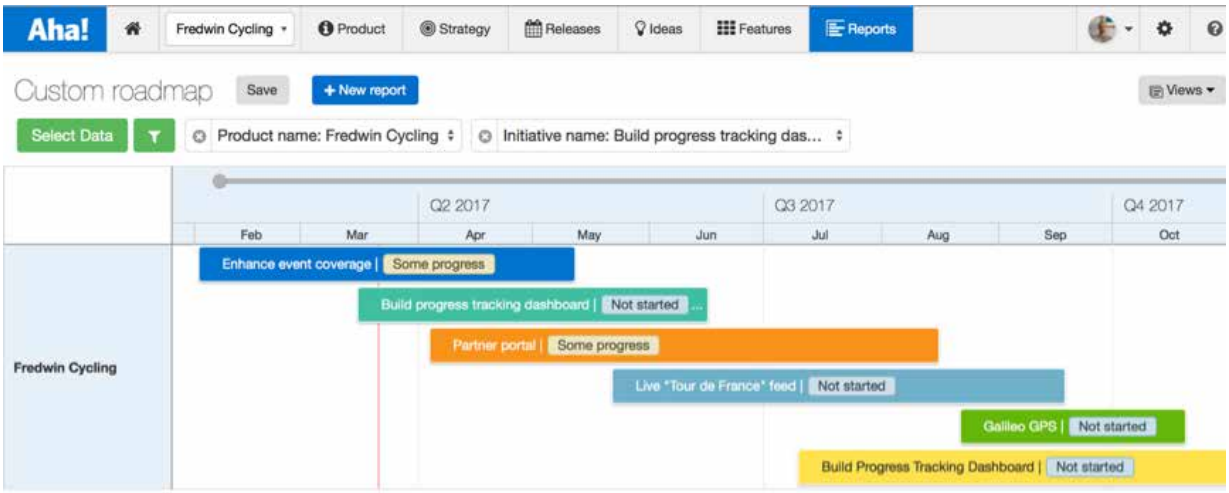
Jede Änderung am Inhalt, Kommentare oder Statusänderungen werden instant übertragen.

Damit erhält der Product Owner eine auf Produktmanagement ausgerichtete Lösung, die deutlich mehr kann, als Jira mit div. Plugins. Das Team kann aber weiterhin in seiner gewohnten Jiraumgebung mit den gleichen Daten arbeiten.

Roadmaps

Roadmaps benötigen insbesondere Planung in die Zukunft. Durch die Beschreibung einer Roadmap bildet der Product Owner die Strategie für die Zukunft sichtbar für alle Stakeholder ab (soferne die Roadmap publiziert wird).

Der große Vorteil liegt darin, dass Änderungen in der Roadmap automatisch durch Änderungen in der Releaseplanung erfolgen. Es ist so möglich, eine jederzeit aktuelle Sicht auf die Roadmap in eine Homepage einzubetten oder als PDF abrufbar zu machen.



Aha! Simplifizierte Roadmap von strategischen Features, wie sie für Kunden publiziert werden kann

Ideenportal

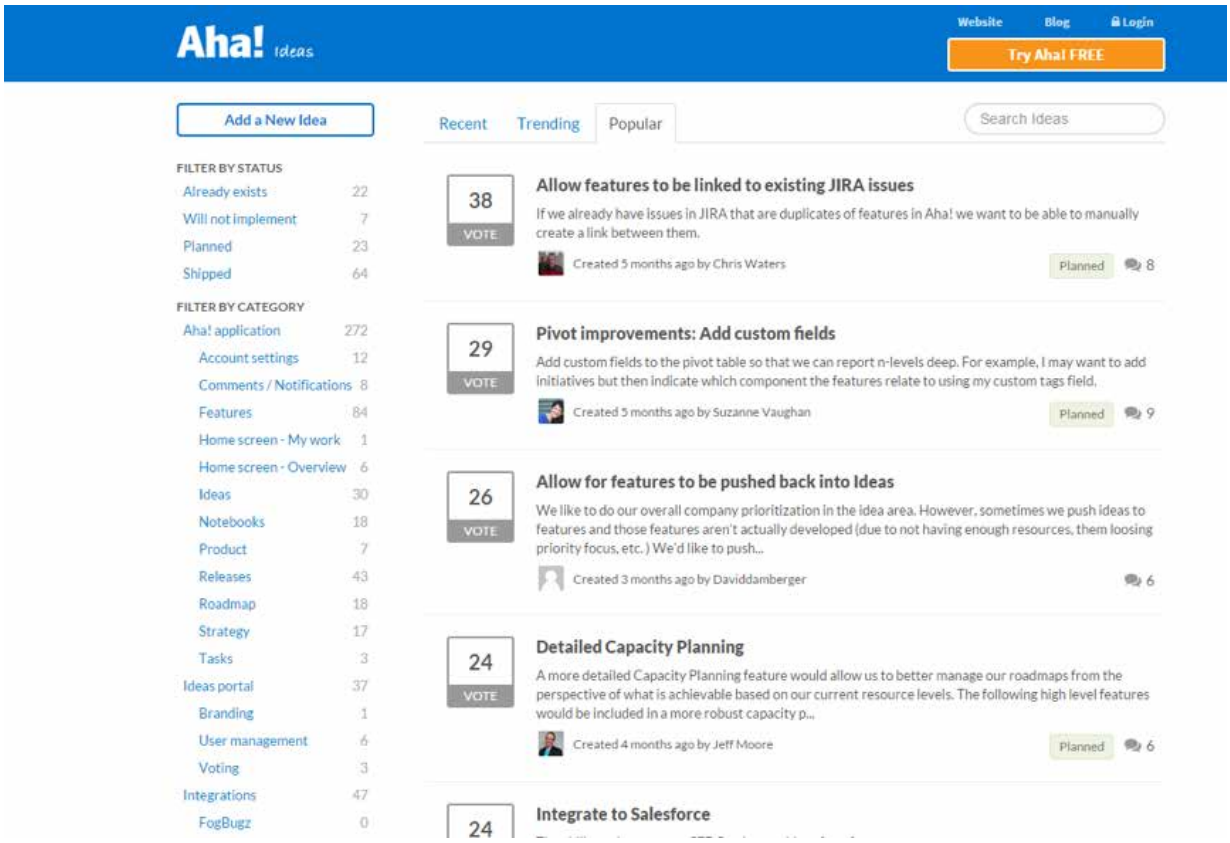
Darüber hinaus bietet Aha! ein eigenes Ideenportal für Kunden. Der Product Owner kann damit über ein anpassbares Portal die Nutzer seiner Kunden auffordern ihre Produktideen einzubringen. Kunden können ihre Ideen auch gegenseitig diskutieren und kommentieren sowie für Ideen anderer Voten.

Dieses Konzept ermöglicht einen direkten Draht zum Kunden, wenn auch nicht alle Ideen umgesetzt werden können, gibt es einen Platz an dem Kunden Ihre Wünsche einbringen können und sie gesammelt werden.

Für den Product Owner stellt das Ideenportal die ideale Kanalisierung der Kundenwünsche dar. Über das Kundenvoting wird transparent, was wichtig und was weniger wichtig ist.

Ideen können direkt in das Backlog übernommen werden, der Status wird danach automatisch im Ideen-portal transparent gemacht und alle Voter für die Idee werden automatisch auf dem Laufenden gehalten.

Titel: Frau im Wasser, Künstler: Nicole Steffler, Technik: Aquarell und Edding



Aha! Ideenportal zur Kanalisierung der Kundenwünsche

Zusammenfassung

Aha! ist ein perfektes System für agile Produktmanagementteams. Es erleichtert die Findung der Produktstrategie, die Releaseplanung und Priorisierung, als auch die Beschreibung der umzusetzenden Anforderungen.

Die Roadmaps als logisches Zusatzergebnis sind ein tolles Addon, ebenso wie das Ideenportal, mit dem große Userzahlen eine direkte Interaktionsmöglichkeit mit dem Produktmanagement bekommen.

Die tiefe Integration in Jira (und viele andere Systeme), ermöglicht es den Teams weiterhin in ihren gewohnten Umgebungen zu arbeiten, aber dennoch Produktmanagement auf einen neuen Level zu heben.

Dabei ist es möglich Aha! an jeden gelebten Prozess anzupassen.■

Wollen Sie Aha! evaluieren, einführen oder mit Ihrem Jira integrieren? Wir unterstützen sie mit unserer Erfahrung und Expertise dabei sehr gerne!

Alexander Vukovic ist SEQIS Gründer und Chief Evangelist.

Er ist erster Ansprechpartner für alle agilen, testmethodischen und testtechnischen Anfragen. In der Praxis arbeitet er als Agile Quality Coach, Berater, Interims-Testmanager, CI-Experte und Lasttester. Mehr als 20 Jahre Beratertätigkeit führten ihn während seiner zahlreichen Projekte in die unterschiedlichsten Branchen und Länder.

Sein persönliches Motto „Es gibt keine Probleme, sondern nur nicht gefunden Lösungen" spiegelt sich in jedem Projekt wider.

UML „Unified Modeling Language“

von *Marlon Gallardo*

„Ein Bild sagt mehr aus als tausend Worte“, diese Aussage ist wohl eine treffende Beschreibung des Unified Modeling Language (UML). Heutzutage benötigt die komplexe und progressive Welt der Softwareentwicklung bestimmte Tools, wie etwa Graphiken und Diagramme, um Systeme, Prozesse und deren Ablaufe verständlicher zu machen. Dabei sind visuelle Erklärungsmodelle hilfreich.

Eines dieser visuellen Tools ist das UML, eine Modellierungsprache, die einen direkten Bezug zu objektorientierter Analyse und Design hat und aus verschiedenen graphischen Elementen, mit Hilfe welcher unterschiedliche Perspektiven eines Systems und dessen Vorgänge dargestellt und beschrieben werden, besteht. Es handelt sich also dabei um eine Reihe von Diagrammen, mit welchen man eine Software detailliert beschreiben kann. Des Weiteren kann UML als Kommunikationsmittel innerhalb eines Projektteams dienen.

Welche Vorteile hat das UML?

- Das Ergründen des Systemverhaltens
- Das Verstehen der Anforderungen an die Software
- Das rechtzeitige Erkennen der Fehler im Lebenszyklus
- Das Präsentieren der Designs
- Die Kommunikation zwischen den Projektbeteiligten

Daraus lässt sich schließen, dass die UML für das Sotfware Testing geeignet ist.

Setzt man sich mit Software Testing nun näher auseinander, wird schnell klar, dass das UML bei dynamischem Test gebraucht wird. Doch worum handelt es sich beim Dynamischen Test? Der Dynamische Test wird das Objekt eines Programms überprüfen und damit Fehlverhalten aufspüren. Das Blackbox Verfahren gehört neben dem Whitebox-Verfahren zum dynamischen Test. Im Folgenden wird allerdings nur auf das Blackbox-Verfahren eingegangen. Das Blackbox-Verfahren: „Der innere Aufbau des Testobjekts ist nicht bekannt und Testfälle werden aus der Spezifikation abgeleitet“.
In diesem Fall könnten die folgenden UML Diagramme angewendet werden:

Anwendungsfalldiagramm (Use Case): Dieses UML stellt eine bestimmte Funktionalität eines Systems dar. Damit kann man die Beziehungen zwischen verschiedenen Funktionalitäten zeichnen bzw. illustrieren. Diese Art von Diagrammen beschreiben hauptsächlich „Nutzer-System-Interaktionen“.

Ein wichtiges Element dieses Diagramms ist der Akteur (Externe Person außerhalb des Systems)

Siehe Bild 1:

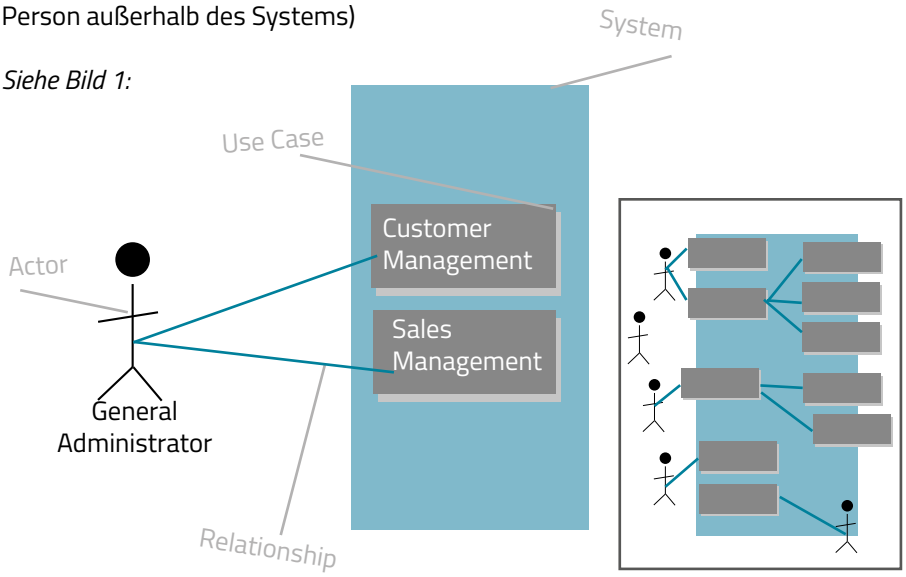


Bild 1
Bildquelle: www.edrawsoft.com/de/uml-introduction.php

Zustandsdiagramme:
Dieses UML beschreibt das Verhalten von Objekten und ihren aktuellen Zustand. Der Begriff „Zustand“ bezieht sich nicht nur auf das Verhalten des Objektes, sondern auf die unterschiedlichen Kombinationen der darin enthaltenen Informationen. Die Verwendung dieses Diagramms ermöglicht es den Benutzern zu einem besseren Verständnis des Systemablaufs zu gelangen.

(siehe Bild 2 unten)

Sequenzdiagramm:
ist eine Art Interaktionsdiagramm, dessen Ziel es ist, das dynamische Verhalten des Informationssystems zu beschreiben, wobei die Reihenfolge der von den Objekten ausgetauschten Nachrichten hervorgehoben wird.
Es ist wichtig zu wissen, dass das Sequenzdiagramm aus der Beschreibung eines Anwendungsfalls (Use Case) erstellt wird.
Ein Sequenzdiagramm hat zwei Dimensionen: Die vertikale Achse repräsentiert die Zeit und die horizontale Achse repräsentiert die verschiedenen Objekte. Die Zeit wird vom oberen Rand des Diagramms nach unten verschoben.

(siehe Bild 3 unten)

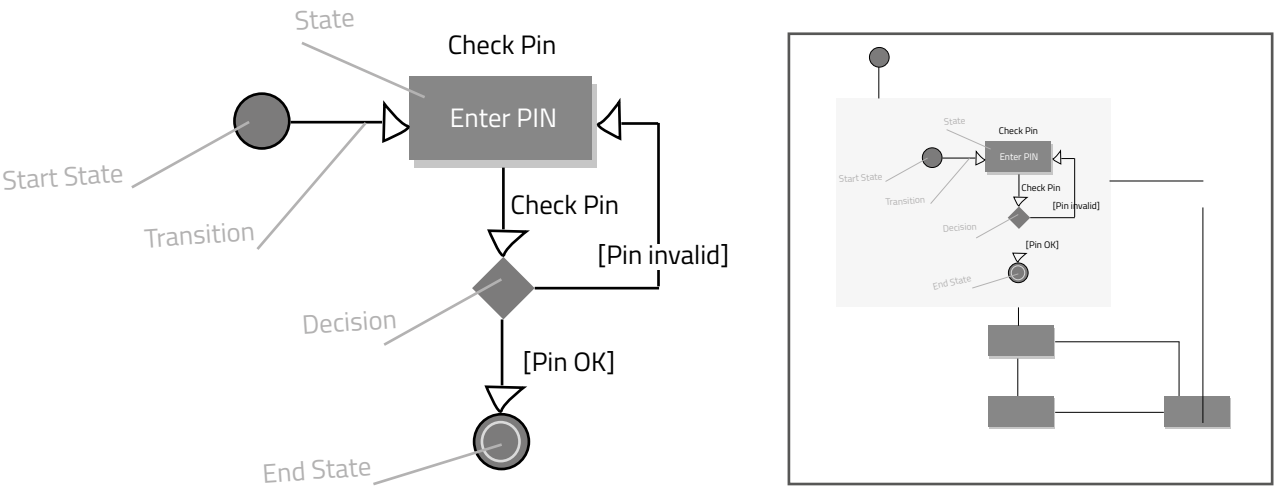


Bild 2: Zustandsdiagramm
Bildquelle: www.edrawsoft.com/de/uml-introduction.php

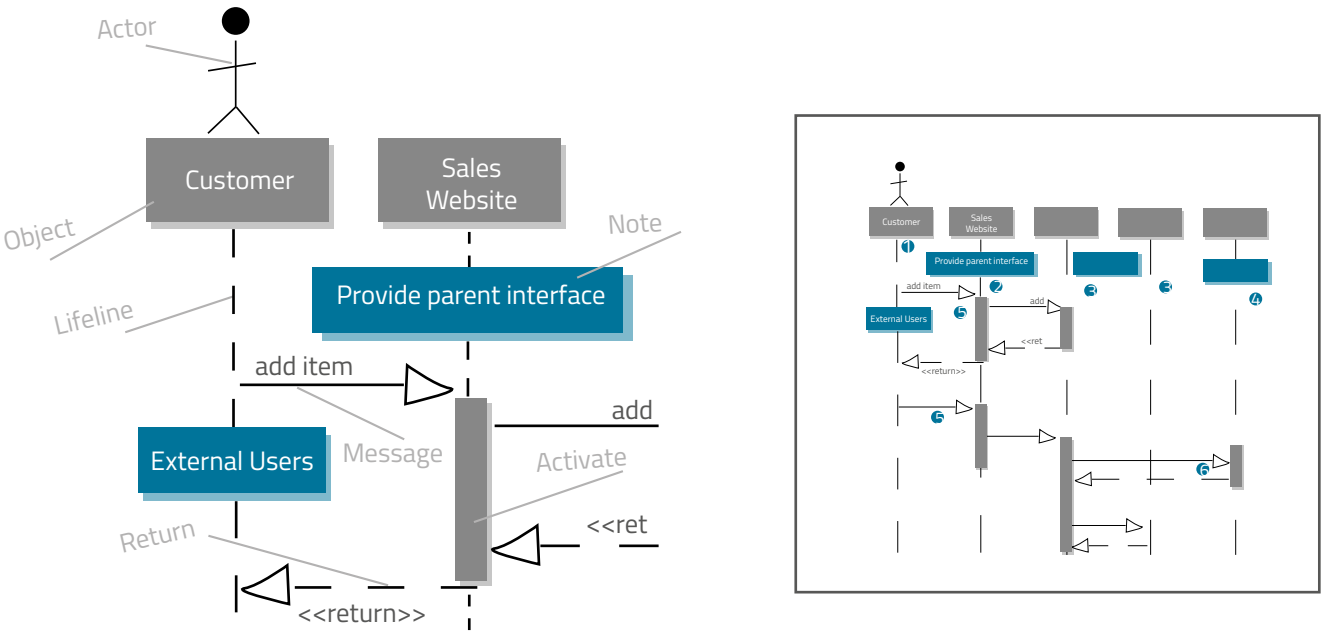
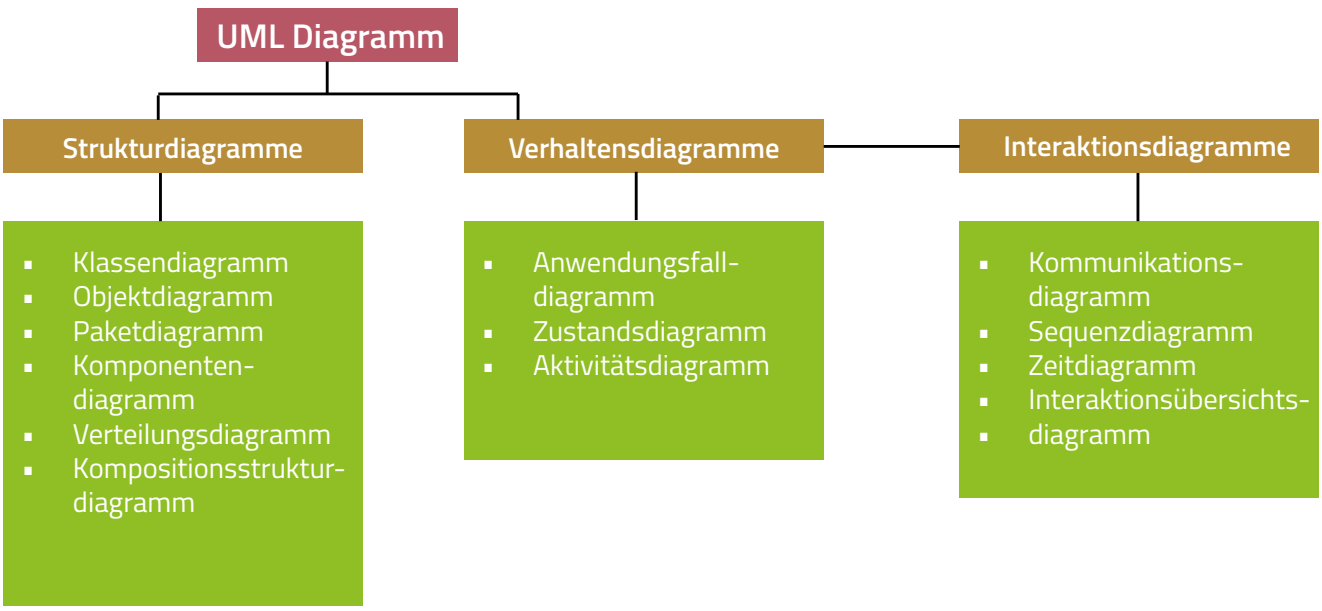


Bild 3:Sequenzdiagramm
Bildquelle: www.edrawsoft.com/de/uml-introduction.php

Es gibt insgesamt 14 UML Diagramme, die nicht nur den oben genannten Zwecken dienen:



Fazit:

Zusammenfassend steht hinter UML die Idee, den Entwurfsprozess so zu organisieren, dass die am Projekt beteiligten Personen diese Entwurfsanalyse sehr leicht verstehen können. Dem Kunden wird also mithilfe von UML eine Entwurfsplanung vorgelegt, die seine Wünsche detailliert beschreibt.

Bei Software Testing ist es grundlegend, alle Details eines Projektes gewissenhaft zu analysieren und zu berücksichtigen. ProgrammiererInnen und TesterInnen sollten die Ideen der Kunden verstehen und diese aktiv erfassen, um deren Projektwünsche realisieren zu können.

Wie am Anfang erwähnt, hat sich die IT- bzw. die Software-Welt so schnell in den letzten Jahren entwickelt, sodass man Tools wie etwa das UML benötigt, um mit diesem Fortschritt mithalten zu können. ■

Referenzen

Basiswissen Softwaretest, Kapitel 5: Dynamische Test, 5.1 Blackbox-Verfahren, Seite 114.
Basiswissen Softwaretest, Kapitel 5: Dynamische Test, 5.1.5 Anwendungs-fallbasierte Test, Seite 145.

Marlon Gallardo ist Consultant bei SEQIS.

Den Fokus legt er dabei unter anderem auf das Software Testing, worin er mehrjährige Erfahrung besitzt. Dabei sind ihm Genauigkeit und die Liebe fürs Detail sehr wichtig.

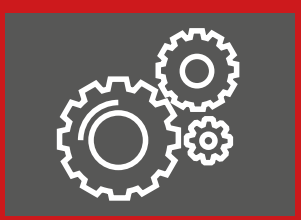
Software Testing ist mehr als ein „Fehler Finden", es ist ein Weg, der zum Ziel die Qualitätssicherung eines Produktes oder Services hat und diesen Weg gehe ich mit Begeisterung und Entschlossenheit.

SEQIS ist der führende österreichische Anbieter in den Spezialbereichen
IT Analyse, Software Test und Projektmanagement.
Beratung, Verstärkung und Ausbildung:
Ihr Partner für hochwertige IT-Qualitätssicherung.

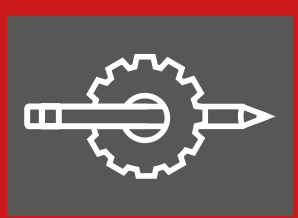
IT ANALYSE
Notwendige Änderungen analysieren und IT-gerecht aufbereiten

CODING
Guten Code schreiben und schlechten Code überarbeiten

TESTING
Probleme durch methodischen Soll-Ist Vergleich erkennen



RELEASE & OPERATE
Reibungsloser Go Live und Betrieb der IT-Lösungen



DEVOPS
Neuerungen abgestimmt mit Entwicklung und Betrieb live setzen

METHODOLOGY & TOOLS
Vorgehensweisen optimieren und auf die richtigen Tools setzen

TRAINING & WORKSHOPS
Mitarbeiter Know-how stärken - standardisiert oder maßgeschneidert

PROJEKT-MANAGEMENT
Verantwortung übernehmen, zielorientiert und pragmatisch agieren

Analysis. Test. Management. Better IT Results.
www.SEQIS.com