SEQIS QualityNews

Analyse, Test und Management

Titel: „Wald im Schmetterlingsmantel", Gruppenbild Kreativgruppe, Technik: Acryl Collage

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 2020 zu präsentieren.

Vielen Dank für das positive Feedback zu unseren letzten Ausgaben zu den Themenbereichen Toolchain und Resilienz. Wir hoffen, wir konnten Ihnen damit interessanten Content zur Verfügung stellen und Sie stellenweise auch gut unterhalten. Über weitere Anregungen, Themenwünsche und Feedback Ihrerseits freuen wir uns.

Auch in dieser Ausgabe finden Sie neben den branchenbezogenen Artikeln auch nicht-technische Bereiche :

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 Legacy Code. Einfach erklärt ist Legacy Code ein Code, der schon lange existiert, stetig erweitert und im Betrieb ist – allerdings auf „alten" Standards basiert.

Auf den folgenden Seiten geben Ihnen unsere Experten einen Einblick in die vielseitigen Aspekte zum Thema Legacy Code.

Wir wünschen viel Lesevergnügen mit unseren SEQIS QualityNews!

Ihre SEQIS Geschäftsleitung

Neulich im Netz: Frauen in der IT

von *Hansjörg Münster*

Der Weltfrauentag wurde kürzlich begangen. Viele Veranstaltungen wurden durchgeführt, die Situation der Frauen in den unterschiedlichen Lebensbereichen thematisiert. Doch wie geht es den Frauen in der IT?

Aus meiner persönlichen Erfahrung als Consultant weiß ich, dass man Frauen im IT Projekten selten trifft. Aber im Projektmanagement, in der Anwenderschulung, im Requirements Engineering und im Testing kenne ich doch einige Frauen, die in der IT ihre Berufung gefunden haben. Aber als IT-Technikerinnen, als Programmiererinnen, DB-Administratorinnen oder Netzwerktechnikerinnen durfte ich in meinem Berufsleben nur sehr wenige kennenlernen.

Ist das wieder mal so eine persönliche Meinung, geboren aus Vorurteilen? Ich wollte es wissen und habe Google um Hilfe gebeten, und einige neue Erkenntnisse gewonnen:

- Wussten Sie, dass im Jahre 2013 der Frauenanteil bei den Maturabschlüssen bei 57,7% und bei den Studienabschlüssen bei 56% lag?
- Allerdings liegt der Frauenanteil der Studienabschlüsse in den technischen Fachrichtungen wie Maschinenbau, Elektrotechnik und Informatik nur bei 10%.
- Wussten Sie, dass die Wissenschaft, die wir heute Informatik nennen, bereits auf das 19. Jahrhundert zurück geht?
- Wussten Sie, dass es schon im 19. Jahrhundert Informatikerinnen gab? Im Jahre 1842 entwarf die britische Mathematikerin Ada Lovelace einen Algorithmus zur Berechnung von Bernoulli-Zahlen mit einer Rechenmaschine? Es tut dabei keinen Abbruch, dass diese, als „Analytical Engine" genannte Rechenmaschine, nie gebaut wurde.
- Wussten Sie, dass der Computer wie wir in heute kennen, auf die ersten Arbeiten in den 1930igern zurückgeht? Und schon damals waren mehr Frauen dabei als heute!
- Wussten Sie, dass im zweiten Weltkrieg das Programmieren fast ausschließlich von Frauen gemacht wurde. Einerseits weil die Männer ja im Krieg waren und andererseits, weil dieses damals an Bürokräfte delegiert wurde.
- Wussten Sie, dass erst in den 1950igern sich der Beruf des Programmierens von einem Frauen- zu einem Männerberuf änderte?
- Wussten Sie, dass sechs Frauen maßgeblich an der Programmierung des legendären ENIAC-Computers beteiligt waren? Der ENIAC war ein Rechner des amerikanischen Militärs zur Erstellung ballistischer Tabellen. Diese sechs Frauen wurden 1997 in einem Festakt geehrt und in die „Women in Technology International (WITI) Hall of Fame" aufgenommen.

Titel: „Stadtgemeinde", Künstler: Robert Schwarz, Technik: Acryl Collage

- Wussten Sie, dass zwei Frauen in leitender Position bei der Entwicklung von COBOL beteiligt waren? Eine von ihnen, Grace Hopper, hat heute noch den Spitznamen „Grandma COBOL" und auf sie gehen Begriffe wie „Bug" und „Debuggen" zurück.

- Wussten Sie, dass es auch heute noch einflussreiche Frauen in der IT gibt? Marissa Mayer war die erste Technikerin bei Google und schaffte es bis zur Vizepräsidentin, was ihr den Spitznamen „Googirl" einbrachte. Später wurde sie dann CEO von YAHOO.

- Wussten Sie, dass es auch in Europa Frauen wichtige Positionen in der IT und Informatik besetzen? Ina Schieferdecker leitete mehrere Fachbereiche am Frauenhofer Institut und seit 2015 das Frauenhofer Institut für offene Kommunikationssysteme und ist heute Professorin an der TU Berlin.

- ... aber wussten Sie, dass in Deutschland der Anteil der Frauen in der IT (abgeschlossene Studien) von 14% im Jahre 1997 auf 7,5% im Jahr 2012 gefallen ist?

Wissen Sie, was mir darüber hinaus zu denken gibt? Die IT ist schon lange kein Beruf mehr, der ausschließlich männliche Nerds, die die Nächte vor dem Computer verbringen, vorbehalten ist.■

Quellen und weiterführende Informationen:

https://www.ocg.at/de/it-frauen

https://de.wikipedia.org/wiki/Frauen_in_der_Informatik

IT Trends und Themen aus den Bereichen Software Test und Business Analyse auf unserem Videoblog!

www.seqis.com/youtube

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.

Begriffsklärung - Was ist ein Legacy System?

von *Tanja Huber*

Ein Legacy System, auf Deutsch „Altsystem", ist gemäß Definition ein Anwendungssystem, Hardware, eine Programmiersprache oder eine Software, welches nicht mehr den aktuellen Stand der Technik entspricht, funktional seinen Zweck aber noch weitgehend erfüllt und weiterhin in Gebrauch ist. In Altsystemen können hierbei Prozeduren und Begriffe vorkommen, welche in aktuellen Kontexten nicht mehr länger relevant oder in Verwendung sind. Das kann zu Verwirrung und Problemen beim Verstehen und Verwenden der Anwendung führen.

Fast jedes Unternehmen hat in irgendeiner Form und Umfang Altsysteme, welche historisch gewachsen sind und aus unterschiedlichsten Gründen über die erwünschte Lebensdauer hinaus weiterverwendet werden. Häufig ist das System betriebsnotwendig und an eine bestimmte (veraltete) Version eines Betriebssystems oder Hardwaremodells gebunden. Und in vielen Fällen handelt es sich um kostenaufwendige Eigenentwicklungen, die für einen bestimmten Zweck entwickelt wurde. Legacy Systeme zeichnen sich durch unzureichende Dokumentation, veraltete Betriebs- und Entwicklungsumgebung, zahlreiche Schnittstellen und hohe Komplexität aus. Diese Eigenschaften, aber auch Faktoren wie der technische und/oder finanzielle Aufwand oder einfach das Risiko von System- oder Datenausfällen können eine Ablösung erschweren.

Beispiele für Legacy Systeme

Nachdem die Raumfahrt ein Bereich ist, bei dem Fehler sehr kostspielig, wenn nicht sogar auch noch gefährlich sein können, finden sich hier oft anschauliche Beispiele, wenn es um Qualität und die Erhaltung von Qualität geht. So kann man sich auch für das Thema „Legacy Systeme" gleich zwei Raumfahrtprogramme der NASA ansehen, die die Vorteile, aber auch Risiken von Altsystemen aufzeigen: Zum einen den „Mars Observer" und zum anderen MAVEN (Mars Atmosphere and Volatile EvolutioN). Der „Mars Observer" ist eine Raumsonde, die entwickelt wurde, um die Marsoberfläche sowie das Klima und Magnetfelder des Planeten zu beobachten. MAVEN ist ebenfalls eine Raumsonde, die die Atmosphäre des Planeten Mars im Rahmen des Mars-Scout-Programms untersucht.

Der **Mars Observer** ist ein Negativbeispiel für die Verwendung von Legacy Systemen. Die Raumsonde wurde am 25. September 1992 erfolgreich gestartet, doch schon am 21. August 1993 wurde die Kommunikation verloren. Das Konzept sah vor, durch Verwendung von beispielsweise standardisierten Satellitenbussen und dem Wiederverwenden des Designs bereits existierender Raumsonden mit geringen Modifikationen, Kosten zu sparen. Im Bericht des Fehlerprüfungsausschuss der NASA ist nachzulesen, dass man sich von diesem Konzept im Laufe des Projektes allerdings zu weit entfernt hat. Als wahrscheinlichste Fehlerquelle wurde ein undichtes Rückschlagventil im Antriebssystem identifiziert. Das Antriebssystem wurde bereits für den Satelliten „Low-Earth Orbit" (kurz LEO) eingesetzt, wobei es für LEO nicht notwendig war, dass das Ventil über einen längeren Zeitraum hinweg dicht blieb – beim Verlust des Mars Observer war es bereits 11 Monate in Betrieb. Das (wieder)verwendete Antriebssystem, welches in diesem Beispiel das Legacy-System darstellt, war demnach für die Mission bzw. die Anforderungen der Mission ungeeignet. NASA hat aus diesem Fehlschlag also unter anderem gelernt, dass Systeme nicht in Umgebungen verwendet werden dürfen, für die sie nicht vorher nachweislich qualifiziert und getestet wurden.

Quelle: NASA, https://www.jpl.nasa.gov/missions/web/mars_observer.jpg

Quelle: NASA, https://www.nasa.gov/sites/default/files/maven_mars_flyover.png

MAVEN als Positivbeispiel verlief innerhalb seines Zeitplans und Budgets, und hatte die volle technischen Leistungsfähigkeit, die geplant war. Man hatte aus vergangenen Fehlern gelernt. So ist im NASA Systems Engineering Handbuch festgehalten, dass ein häufig übersehener Bereich jener ist, der mit der Modifikation von „Heritage" (engl.: das Erbe) -Systemen verbunden ist, die in verschiedene Architekturen integriert sind und in anderen Umgebungen arbeiten als die, für die sie entworfen wurden.

Legt man die Erkenntnisse aus diesen beiden Beispielen also wieder auf Software um, so kann eine an die geforderten Anforderungen gut angepasstes Legacy-System erfolgreich und kostensparend eingesetzt werden. Ein nicht angepasstes Legacy-System birgt allerdings Risiken, die beispielsweise kritische Folgen für das Unternehmen haben kann. Die Entscheidung, ob die Verwendung von Legacy-Systemen empfehlenswert ist, kann man also nicht pauschal beantworten. Es ist von Fall zu Fall zu entscheiden, ob die Vorteile die Risiken aufwägen, beziehungsweise ob man für die Vorteile die Risiken in Kauf nehmen will.■

Quellen und weiterführende Informationen:

„Propulsion Lessons Learnd from the loss of Mars Observer", Carl S. Guernsay (weitere Infos müsste ich nochmals mittels Google herausfinden)

„Heritage Technologies in Space Programs - Assessment Methodology and Statistical Analysis", Dipl. -Ing. Univ. Andreas Makoto Hein, 27.06.2016

https://www.researchgate.net/publication/310478449_Heritage_Technologies_in_Space_Programs_-_Assessment_Methodology_and_Statistical_Analysis

https://trs.jpl.nasa.gov/handle/2014/36825

https://mars.nasa.gov/mars-exploration/missions/mars-observer/

https://www.sciencedirect.com/science/article/pii/S1474667017588953

Tanja Huber, ist Consultant.

Zusätzlich zu ihrer Erfahrung in den Bereichen Testfallerstellung, Testdurchführung und Testunterstützung ist sie auch in den Bereichen Projekt- und Qualitätsmanagement tätig.

Beim Herangehen an neue Aufgaben ist ihr eine gewissenhafte Vorgehensweise und kreative Aufgabenlösung wichtig. Ebenso legt sie größten Wert auf hochwertige Qualitätssicherung.

Was ist Legacy Software - und warum hat sie einen schlechten Ruf?

von *Josef Falk*

Haben Sie schon einmal geerbt? Dann haben Sie das vermutlich durchaus als angenehm empfunden und haben Ihr Erbe nicht als „Altlast" bezeichnet. In der IT ist das anders. Da empfindet man ein Erbe – und nichts anderes bedeutet das Wort „Legacy" – überwiegend als etwas Negatives, mit dem man irgendwie umgehen muss. Und es wäre besser, so etwas nicht zu haben. Warum ist das so? Und was genau ist eigentlich „Legacy Software"?

Quelle: Bild von Alexander Lesnitsky auf Pixabay

Einer der Punkte, die in jeder Definition von „Legacy Software" implizit vorkommt, ist, dass es sich um „alte" Software handelt. Nun ist Software aber nichts Materielles. Warum unterliegt sie dennoch einem Alterungsprozess?
Das sind die beiden Fragen, die in diesem Artikel im Folgenden behandelt werden:

- Was ist „Legacy-Software"?
- Warum wird Software „alt"?

Was ist „Legacy-Software"
Wenn man von „Legacy-Software" spricht, meint man Software, von der man das Gefühl hat, dass man sie lieber nicht hätte und sie trotzdem aus irgendwelchen Gründen nicht einfach los wird.

Hier sprechen wir aber von einem vagen Gefühl, das nicht den Kriterien einer Definition entspricht. Wir wollen uns im Folgenden einige in der Literatur vorhandene Definitionen ansehen, ohne uns aber letztlich auf eine festzulegen.

Michael Feathers bezeichnet **„Code ohne Tests"** als Legacy [Fea04]. Grund dafür ist, dass es riskant ist, Software zu ändern, für die keine automatisierten Tests vorhanden sind. Diese Definition trifft sicher einen wahren Kern. Auf der anderen Seite hieße das aber auch, dass brandneue Software „Legacy" wäre, wenn keine automatisierten Tests für sie geschrieben wurden. „Software ohne Tests" ist also eher Software, die keinen hohen Qualitätsansprüchen genügt, „Legacy" ist sie nicht notwendigerweise.

Adam Tornhill definiert in [Tor18] Legacy als **„Code mit Qualitätsmängeln, den wir nicht selbst geschrieben haben".** Auch diese Definition scheint den Punkt nicht genau zu treffen. Der Autor von Software ist nicht unbedingt entscheidend dafür, ob es sich um Legacy-Software handelt. Wenn man schon lange genug in der IT-Entwicklung tätig ist, kann es durchaus vorkommen, dass man einst Software geschrieben hat, die heute Legacy ist. Und auch die Qualitätsmängel sind nicht das Relevante. Qualitätsmängel hat oft auch ganz neuer Code – und manch alter Code, den wir zu Recht als Legacy bezeichnen, hat solche nicht – zumindest hatte er sie nicht zum Zeitpunkt, als er geschrieben wurde.

In [IDG18] spricht man von Legacy-Systemen häufig im Zusammenhang mit **Mainframes**. Es ist wohl aktuell derzeit tatsächlich so, dass die als problematisch betrachteten Legacy-Systeme auf Mainframes laufen. Mit fortschreitender Zeit werden aber auch mehr und mehr andere Systeme (z. B. Client Server) die Kriterien für Legacy-Systeme erfüllen.

Nicht ganz ernst gemeint sind wohl Definitionen wie: Legacy, das ist Code, **den die Entwickler nicht mögen** oder Legacy sei Software **ab dem zweiten Sprint.**

Eine Definition, die die Sachlage sehr gut beschreibt, ist jene auf Wikipedia [WikiLC]: Legacy ist „Code mit **Abhängigkeiten zu nicht mehr unterstützten Betriebssystemen oder anderen Technologien"**. In der Tat ist es fast

immer die veraltete Programmiersprache, die nicht mehr unterstützte Datenbank oder das Betriebssystem, die den Umgang mit alter Software problematisch machen.

Warum altert Software?
Egal, welche der Definitionen man heranzieht, sie haben alle etwas mit dem Alter der Software zu tun. Legacy-Software wird als schlecht angesehen, weil sie alt ist. Warum ist das aber überhaupt ein Problem? Software ist ja nicht etwas Materielles, das etwa einem Korrosions- oder einem anderen Alterungs-Prozess unterliegt. Wieso wird Software trotzdem alt?
Es gibt unterschiedliche Gründe, die dazu führen, dass Software altert. In [Sne19] werden sechs Gründe der Softwarealterung genannt:

- **der Software-Alterungsprozess:** Software altert, wenn sie nicht mehr mit ihrer Umgebung übereinstimmt. Hardware ändert sich. Neue Software, wie Betriebssysteme, Datenbanksysteme und Programmiersprachen erfordern neue Schnittstellen. All diese Änderungen sind unabhängig von der Qualität der Software, unabhängig davon, wie gut sie konstruiert und wie sorgfältig sie implementiert ist.

- **der Weg der Fehlentscheidungen:** Fehlentscheidungen fallen oft schon in die Anfangszeit der Software. Anwender entscheiden sich für ein unreifes Produkt oder steigen in eine Technologie ein, die sich nicht durchsetzt. Auch derartige Fehlentscheidungen lassen sich kaum verhindern. Es ist nicht vorauszusehen, ob die tolle neue Technologie, von der heute jeder spricht, auch in wenigen Jahren noch von Bedeutung sein wird. Wer kann etwa mit Sicherheit sagen, ob sich heute aktuelle Programmiersprachen wie Clojure oder Node-JS dauerhaft durchsetzen?

- **der Weg des Personalverlusts:** Es kommt auch vor, dass eine Technologie zwar weiter unterstützt wird, dass aber die Wissenden darüber verloren gehen. Das ist einerseits häufig der Falle bei stark gehypten Technologien, die sich letztendlich dann doch nicht durchsetzen. Aber auch Technologien, die einmal Industriestandard waren – wie es etwa bei der Programmiersprache COBOL der Fall war – teilen dieses Schicksal, wenn die Entwicklung fortschreitet – und die entsprechenden Ausbildungen nicht mehr angeboten werden oder auch einfach nicht mehr attraktiv sind.

- **der Softwareverschleiß:** Obwohl Software nichts Materielles ist, unterliegt sie doch einem Verschleiß. Durch andauernde Änderungen, die durchaus notwendig sind, entfernt sie sich

Was verbirgt sich hinter diesem QR Code?

Titel: Vorzimmer, Künstler: Katharina Stockreiter & Christian Reithofer, Technik: Acryl auf Leinwand

immer weiter von ihrer Urarchitektur, sodass die ursprünglich geltenden Prinzipien ständig verletzt werden. Dadurch wird die Software immer schlechter lesbar und weitere Änderungen werden immer schwieriger.

- **die Komplexitätssteigerungen:** Wenn immer mehr Funktionalität in die Software hineingepackt wird, anstelle sie in andere Systeme auszulagern, hat das ähnliche Folgen wie beim Softwareverschleiß: Die Software verliert die (wirtschaftliche) Wartungsfähigkeit.

- **der Qualitätsverfall:** Von Qualitätsverfall spricht man dann, wenn Änderungen und Fehlerkorrekturen unüberlegt und schlampig durchgeführt werden. Auf diesem Weg wird die Software immer brüchiger.

Nur gute Software wird Legacy
Oft wird die Frage gestellt, was man denn tun könne, damit heute geschriebene Software nicht Legacy wird. Die Antwort ist relativ einfach: Wenn Sie nicht wollen, dass Ihre heute geschriebene Software nicht Legacy wird, dann schreiben Sie schlechte Software.

Es lässt sich nicht verhindern, dass neue Programmiersprachen auf den Markt kommen – auch dem heute allgegenwärtigen Java wird es nicht anders ergehen. Genauso wird es mit den heutigen Betriebssystemen, Datenbanken, Hardware-Komponenten sein. Qualitativ hochwertige Software kann diese Umwälzungen überstehen. Von schlechter Software wird man sich im Zuge dieser Änderungen leichter trennen.

Darum: Wenn auch Sie sich mit Legacy-Software herumschlagen müssen, so mag es bei allem Ärger und aller Mühsal ein Trost sein: Nur gute Software wird Legacy.■

Quellen und weiterführende Informationen:

[Fea04] M. Feathers, Working Effectively With Legacy Code, Prentice Call, 2004

[IDG18] IDG Studie zur Modernisierung von Legacy-Systemen 2018

[Lop18] E. Lopian, Defining Legacy Code, DZone, 15.5.2018, siehe: https://dzone.com/articles/defining-legacy-code

[Sne19} H. Sneed, Was tun mit der Altsoftware – entsorgen, konvertieren oder wiederverwenden?, in ObjektSpektrum 06/2019

[Sta19] G. Starke, Vermächtnis in kleinen Schritten kontinuierlich fortentwickeln, Legacy ist keine Krankheit, in ObjektSpektrum 06/2019

[Tor18] A. Tornhill, Software Design X-Rays, Pragmatic Programmer, 2018

[Web19] M. Weber, Vermeidung der Legacy-Falle, Ein Erfahrungsbericht am praktischen Beispiel, in: ObjektSpektrum 06/2019

[WikiLC] Legacy Code, siehe: https://en.wikipedia.org/wiki/Legacy_code

Mag. Josef Falk ist IT Analytiker.

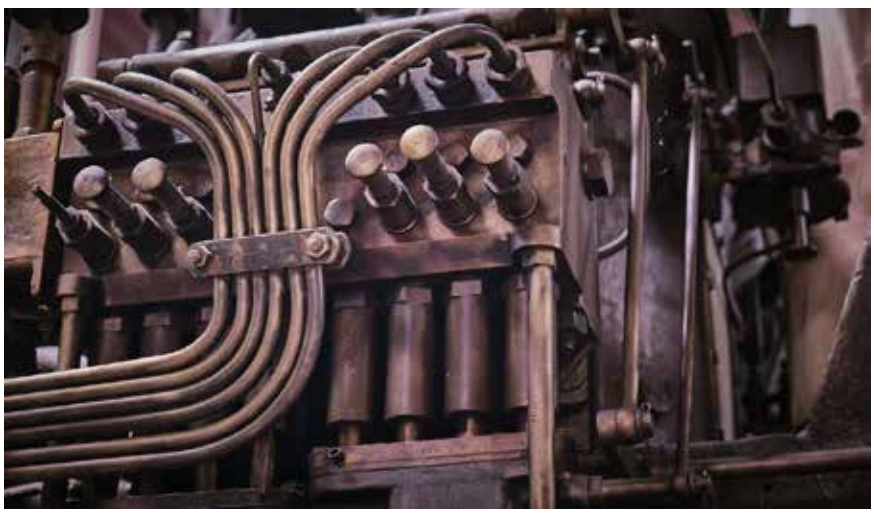
Seit dem Abschluss seines Studiums der Betriebswirtschaftslehre in Wien gestaltet er Lösungen in den unterschiedlichsten Fachbereichen – und ist dabei Mittler zwischen Fachbereich und IT-Entwicklung.

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).

Strategien im Umgang mit Legacy Code

von Stefan Ladstätter

Für Entwickler ist Legacy Code meist ein Synonym für „schlechten Code" – schwer zu lesen, schwer zu verstehen und schwer zu warten. Wir wollen kurz beleuchten, wie es dazu kommt und welche Strategien wir anwenden können, um Legacy Code zu transformieren.



Quelle: Image by Daniel Kirsch from Pixabay

If it ain't broke, don't fix it.
Niemand beginnt ein Softwareprojekt mit dem Vorsatz, schlechten Code zu schreiben. Code ist jedoch ständiger Änderung unterworfen. Neue Features kommen hinzu, Fehler werden behoben, die Performance soll optimiert werden oder Refactoring durchgeführt werden. Dabei soll existierendes Verhalten der Software – der weitaus größere Teil des Codes – nicht beeinträchtigt werden.

Da jede Änderung ein Risiko birgt, ist die Herangehensweise von Entwicklerteams traditionell eher konservativ. Möglichst wenig existierender Code soll angetastet werden, um das Risiko von ungewünschten Nebenwirkungen gering zu halten. Der Nachteil: Notwendige strukturelle Änderungen werden nicht vorgenommen, es häuft sich eine „technische Schuld" an, welche die Angst vor Änderungen weiter steigen lässt. Jeder Eingriff in die Code-Basis wird zu einem Drahtseilakt ohne Sicherheitsnetz, und selbst nach Einarbeitung in den Code bleibt oft nur Hoffen und Beten, dass nichts kaputtgeht.

Legacy Code = Code ohne Tests
Dieser Teufelskreis zeigt, dass man Legacy Code auch so definieren kann, wie dies Michael Feathers in seinem 2004 erschienenen Standardwerk „Working Effectively With Legacy Code" tut: **Legacy Code ist Code ohne Tests.** Denn unabhängig davon, wie elegant oder wie performant Code geschrieben ist, nur mit Tests kann das Verhalten von Code zeitnah und mit Zuversicht verändert und evaluiert werden.

Um sicherzustellen, dass die Software nach einer Änderung weiterhin wie erwartet funktioniert, werden Regression-Tests durchgeführt. Solche Tests sind wichtig, aber meist recht zeitintensiv, was dazu führt, dass sie nicht nach jeder kleinen Änderung durchgeführt werden, sondern in periodischen Abständen. Das macht es schwierig, die Ursache für Fehlerzustände zu lokalisieren.

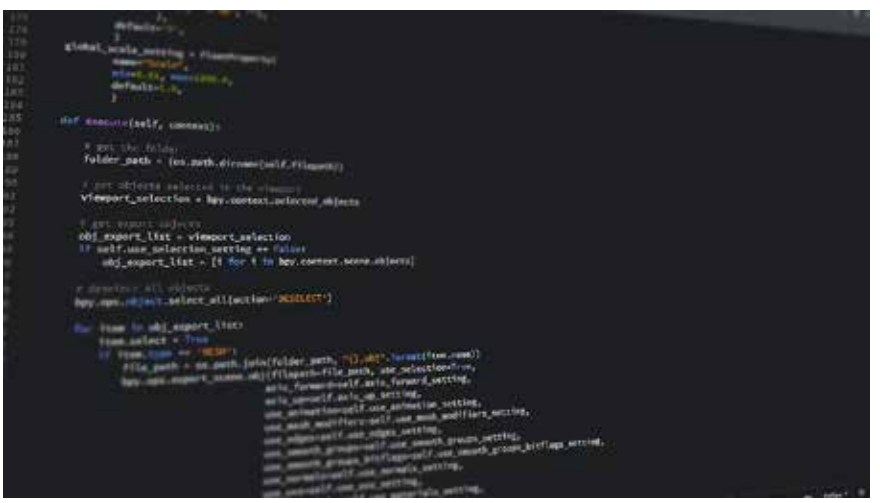
Um beim Testen die Feedback-Loops möglichst kurz zu halten und die Fehlerlokalisation zu erleichtern, bieten sich daher Komponenten- bzw. Unit Tests an. Dazu werden kleinste Software-Bausteine in ein Testgerüst gepackt, welches die umliegende Software simuliert. Die Vorteile liegen auf der Hand: Durch die kurze Laufzeit (oft wenige Millisekunden) gibt es keinen Grund, den Test auf später zu verschieben. Durch das Testen kleinster Code-Segmente ist die Fehlerlokalisation trivial und wir können sicher sein, dass wir eine hohe Testüberdeckung erzielen.

Der Trick zum Vermeiden von Legacy Code ist daher, Unit Tests zu schreiben, und zwar vor dem Verfassen des dazugehörigen Codes. So kann man Funktionalität in einen festen Testrahmen gießen. Jeder Änderung geht das Ändern bzw. Verfassen der zugehörigen Tests voraus. Nur so können wir mit Bestimmtheit sagen, dass nach einer Änderung keine unerwarteten Nebeneffekte auftreten werden. So lässt sich Legacy Code elegant vermeiden. Der Aufwand erscheint zunächst unrealistisch hoch, verblasst jedoch beim Vergleich zu dem Aufwand, der nötig ist, um Änderungen in Legacy Code vorzunehmen.

Unit Tests
Unit Tests haben eine Laufzeit gemessen in Millisekunden. Ein Test erfüllt nicht die Kriterien eines Unit-Tests, wenn er eines oder mehrere der folgenden Merkmale aufweist:

- Er spricht mit einer Datenbank.
- Er kommuniziert über ein Netzwerk.
- Er interagiert mit dem Dateisystem.
- Man muss erst spezielle Dinge durchführen, um ihn auszuführen (z.B. Konfigurationsdateien editieren).

Solche Tests sind ebenfalls gut zu haben oder zu schreiben, aber diese Merkmale verhindern, dass sie schnell durchgeführt werden können, was wiederum dazu führt, dass sie zumeist nicht unmittelbar nach jeder Änderung durchgeführt werden.



Quelle: Image by Johnson Martin from Pixabay

Breaking up is hard to do
Wir haben gelernt: Vor jeder Änderung muss es bereits die dazugehörigen Tests geben. Dies stellt uns vor ein Dilemma, denn um Tests für vorhandenen Legacy Code zu ermöglichen, müssen wir zunächst erst einmal den vorhandenen Code ändern.

Das größte Problem hierbei sind Abhängigkeiten. Hängt beispielsweise eine Legacy-Klasse direkt von etwas anderem ab, was man nicht einfach in einem Unit Test benutzen kann, macht es eine Änderung sehr riskant.

Abhängigkeiten stellen daher eines der schwierigsten Probleme in der Software-Entwicklung dar, und die Arbeit mit Legacy Code ist vorrangig damit beschäftigt, Abhängigkeiten aufzubrechen, um das Vornehmen von Änderungen zu erleichtern.

Der Änderungsprozess für Legacy Code
Das Ziel bei der Arbeit mit Legacy Code ist das Vornehmen von Änderungen, die sowohl einen Mehrwert darstellen als auch mehr und mehr Teile des Systems in ein Testgerüst spannen. Im Verlauf der Zeit werden aus den Inseln von getesteter Code-Basis immer größere Landmassen und schlussendlich zu Kontinenten von Code, der durch Tests abgedeckt ist.

Für diesen Vorgang haben sich die folgenden Schritte bewährt:

1. Änderungspunkte identifizieren
Welche die neuralgischen Stellen im Code sind, die verändert werden sollen, hängt immer von der Softwarearchitektur ab. Um sich mit der Architektur vertraut zu machen, haben sich diverse Techniken bewährt, wie dem Erstellen eines Code-Organigramms oder dem sogenannten Wegwerf-Refactoring. Hier arbeitet man an einer Kopie des Quellcodes, deren Codestruktur man ohne Rücksicht auf Verluste und in Bausch und Bogen ändern darf, inklusive dem Löschen von unbenutztem Code. Es ist überraschend, welche Erkenntnisse bei diesem Vorgang in kürzester Zeit gemacht werden. Das Programm ist danach zwar nicht lauffähig, aber das ist auch nicht Zweck der Übung.

2. Testpunkte identifizieren
In Legacy Code ist es oft nicht ganz einfach herauszufinden, wo man die Tests schreiben soll, um eine Änderung durchzuführen. Dazu müssen wir eine Effekt-Analyse durchführen und Code-Teile aufspüren, die von unseren Änderungen betroffen sein könnten (Methoden, die unsere Methode aufruft oder auf Werte zugreift, die unsere Methode modifiziert; alles, was wiederum jene Methoden aufruft; Super- und Subklassen, die unsere Instanz-Variablen oder jene Methoden verwenden; Parameter jener Methoden und ob sie von unseren Änderungen betroffen sein könnten, sowie globale Variablen und statische Daten, die von unserer oder den umliegenden Methoden modifiziert werden). Diese gilt es dann in Ablaufdiagrammen zu skizzieren, auf deren Basis dann die Testpunkte gesetzt werden können.

3. Abhängigkeiten aufbrechen
Um Unit Tests zu schreiben, müssen wir unseren Code in unabhängige, kleine Stückchen teilen, die wir ein Testgerüst spannen können. Hier wird unser Legacy Code Dilemma am deutlichsten: Wo sind die Tests, die uns zeigen, ob wir durch das Aufbrechen von Abhängigkeiten neue Probleme verursachen? Das bedeutet, dass wir bei diesem Schritt besonders sorgsam und diszipliniert vorgehen müssen: Keine funktionalen Änderungen, reines Refactoring. Änderungen auf das allernotwendigste reduzieren (z.B. keine Optimierung so ganz nebenbei). Hier kann Pair

Programming helfen. Das Bearbeiten von Legacy Code ist vergleichbar mit einem chirurgischen Eingriff – und Ärzte operieren schließlich auch nie allein.

4. Unit Tests schreiben

Die meisten Leute verbinden Tests mit dem Aufspüren von Fehlern, meist in manuell durchgeführten Tests, denn das Schreiben von automatisierten Tests für das Aufspüren von Fehlern in Legacy Code erscheint wenig effizient. Das Problem: Manuelle Tests sind zeitraubend und zermürbend, daher werden sie nicht nach jeder Änderung durchgeführt. Abgesehen davon ist es keine Herausforderung, Bugs in Legacy Code zu finden, jedoch ist das nicht das zielführendste Investment von Zeit und Ressourcen.

Viel effizienter ist es, durch den Einsatz von automatisierten Unit Tests neue Fehler zu vermeiden. Unit Tests spezifizieren erwünschtes Verhalten, das Ziel, das es zu erreichen gilt. Das Aufspüren von Bugs ist ein Nebeneffekt – Fehler treten erst dann auf, wenn eine Änderung eine unerwartete und unerwünschte Auswirkung hat.

In Legacy Code gibt es noch keine Tests, die sicherstellen, dass das gewünschte Verhalten bewahrt wird, wenn wir Änderungen vornehmen. Dazu müssen wir in einem ersten Schritt das tatsächliche Verhalten des betrachteten Legacy Codes herausfinden und dafür Tests schreiben, bis wir den Code, den wir ändern wollen, verstanden haben und dass unser Refactoring das Verhalten nicht beeinträchtigt. In einem zweiten Schritt analysieren wir, ob die neu geschriebenen Tests auch jenes Verhalten abdecken, das wir ändern wollen und fügen im Bedarfsfall auch dafür Tests hinzu.

5. Änderungen am Code vornehmen und Refactoring

Haben wir es bis hierher geschafft, ist der Rest ein Kinderspiel. Mithilfe von Test-Driven Development (TDD) bauen wir die neue Funktionalität ein und testen sie, anschließend führen wir Refactoring durch. Befolgen wir diesen Ablauf Tag für Tag, können wir unsere Code-Basis über kurz oder lang aus dem Legacy-Zustand befreien.■

Der Änderungsprozess für Legacy Code

Quellen und weiterführende Informationen:

Michael C. Feathers, Working Effectively with Legacy Code, 2004, Prentice Hall.

Stefan Ladstätter, ist Consultant.

In seiner mehr als 20-jährigen Tätigkeit als Entwickler, Requirements Engineer, Usability-Spezialist und Projektmanager hat er Einblick in die unterschiedlichsten Aspekte der Umsetzung von IT-Projekten gewonnen und kann dabei auf einen tiefen Erfahrungsschatz im klassischen und agilen Umfeld vorweisen.

Er sieht erfolgreiches Projektmanagement als Dienst an allen Projektbeteiligten – Auftraggeber, Entwicklungsteam und Anwender.

Legacy Applikationen - Ab wann wird Legacy zum Risiko?

von Alexander Vukovic

Bei Autos ist es ziemlich klar: Ein gut gepflegter Oldtimer kann 30 Jahre und mehr seinen Dienst tun. Vorausgesetzt er wird gut gewartet. Schlecht gepflegte Fahrzeuge müssen jedoch schon mit 10 Jahren oder weniger aus Sicherheitsgründen aus dem Verkehr gezogen werden.

Aber wie ist das bei Software? Unterliegt Software dem gleichen Verschleiß und dem gleichen Alterungsprozess wie ein Auto oder ein Reifen?

Würde man Software z. B. im Rahmen eines Escrow-Vertrages (~ spezifische Form einer rechtlichen Hinterlegung) von heute in einem Safe hinterlegen und sie in 20 Jahren wieder öffnen, wäre sie mit sehr hoher Wahrscheinlichkeit, genauso wie sie abgespeichert wurde, immer noch am Datenträger vorhanden.

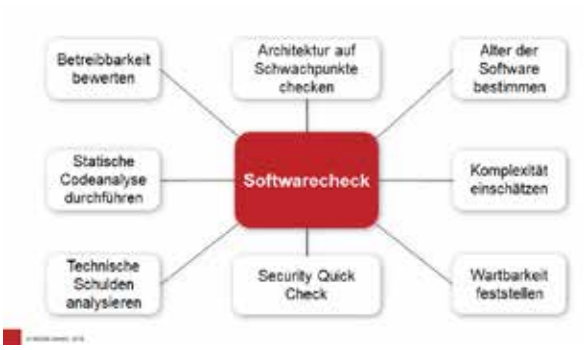
Dennoch ist sie massiv gealtert.
Der Alterungsprozess bei Software erfolgt im Kontext zu ihrer Umgebung. Legacy Software und Legacy Code entsteht also dadurch, dass sich einerseits niemand um sie kümmert und andererseits sich die Umgebung weiterentwickelt und die Software stagniert.

Als Eigentümer und/oder Betreiber einer Software sollten Sie sich daher regelmäßig fragen: Kriegt Ihre Software so noch ein „Pickerl"? Können Sie die Software mit vertretbarem Risiko so noch weiter betreiben? Womit haben Sie zu rechnen, wenn Sie dringend und kurzfristig Veränderungen an der Software vornehmen müssen? Wie viele Ihrer Mitarbeiter oder Partner können die Software überhaupt noch mit vertretbaren Aufwand verändern?

SEQIS hat mit dem **„Application Quality Quick Assessment" (kurz AQQA)** eine Art Pickerlüberprüfung für Ihre Bestandssoftware entwickelt.

Zum kalkulierbaren gedeckelten Maximalaufwand von 5 Personentagen überprüfen unsere Experten Ihre Software und stellen einen detaillierten Prüfbericht zusammen, der als Entscheidungsgrundlage für den weiteren Betrieb oder etwaige kurzfristig notwendige Mängelbeseitigungen herangezogen wird.

Das Application Quality Quick Assessment betrachtet dabei die unterschiedlichsten Aspekte mit 360 Grad auf Seite der Software und des Codes.



Ihre Software wird dabei einerseits auf die langfristigen inneren Werte wie Betreibbarkeit, Architektur, Wartbarkeit und Code Qualität geprüft. Andererseits werden aber auch das tatsächliche Alter und der aktuelle Zustand z. B. im Bereich Security und Softwareaktualität festgestellt.

Läuft Ihre Software z. B. in einer Java VM und verwendet dabei diverse bekannte Libraries wie Hibernate oder SmartGWT, dann ist die Frage, ob diese Frameworks und Java immer laufend aktualisiert wurden. Ist z. B. die eingesetzte Hibernate Version bereits 10 Jahre alt, dann ist die Software als Gesamtes dadurch massiv gealtert. Die Aktualisierung hätte eigentlich laufend durchgeführt werden müssen, dies nicht zu tun erzeugt technische Schulden. Sehr häufig weiß man, was zu tun wäre, hat aber keine „Zeit".

Technische Schulden sind unterlassene Änderungen, aber bewusst oder unbewusst nicht gemacht werden. Durch die Vernachlässigung sind aber auch laufend Zinsen fällig. Ihre Software hat dann z. B. ein massiv höheres Security Risiko. Denn die Security Lücken für Hibernate der letzten 10 Jahren sind öffentlich zugänglich. Ein Angreifer, der auf eine 10 Jahre alte Hibernateversion trifft, bekommt damit gleichzeitig einen Katalog, welche Angriffsvektoren sicher funktionieren gratis aus dem Netz.

Auch die erschwerten Änderungen, die nicht (mehr) Kompatibilität zu einer Java-Version oder anderen Frameworks erzeugt massiven zusätzlichen Aufwand, also Zinsen, die durch die technischen Schulden entstehen. Unser Check prüft und analysiert also das Alter und den aktuellen Stand der technischen Schulden.

Sind die Aufwände zum Bereinigen der technischen Schulden größer als jene für eine Neuentwicklung, sollte man über die Neuentwicklung nachdenken.

Das Alter der Software steigt auch dadurch, dass die Software in einer alten, nicht mehr gängigen, Programmiersprache geschrieben wurde, wie z. B. Cobol, Progress oder Assembler. Sie wurde zwar zum Zeitpunkt der Entwicklung vollkommen korrekt und nach Lehrbuch entwickelt, heute würde man das aber nicht mehr so machen und junge Entwickler hätten Schwierigkeiten sich in die Denkweisen von vor 20 Jahren hineinzudenken. Code mit schlechter Code Qualität erzeugt massive Mehraufwände und erhöhtes Fehlerpotential, wenn er dann doch geändert werden muss, meist durch jemanden, der keinerlei Erfahrung und Bezug zu der Software hat.

Das Application Quality Quick Assessment überprüft z. B. ob bei Änderungen diese auch korrekt durch entsprechende Unit Tests für Legacy Code abgesichert werden. Dabei gibt es spezielle Techniken, die nicht das Ziel haben, den gesamten Code unter Abdeckung zu bringen, sondern die Änderung so weit wie möglich abzusichern, so dass diese keinerlei negativen Nebeneffekte auf den funktionierenden Bestandscode hat.

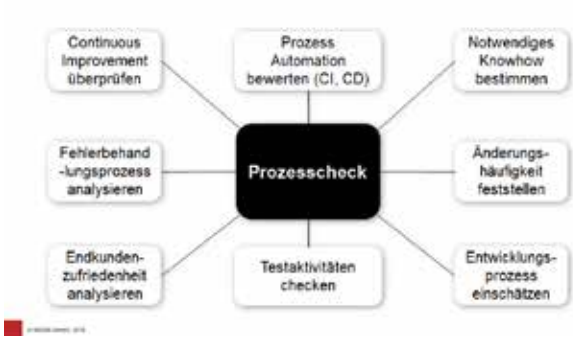
> Code without tests is bad code. It doesn't matter how well written it is; it doesn't matter how pretty or object-oriented or well-encapsulated it is. With tests, we can change the behavior of our code quickly and verifiably. Without them, we really don't know if our code is getting better or worse.

Aus "Working effectively with Legacy Code (Robert C. Martin Series)"

Erfolgt die Entwicklung bei Ihnen im Haus, durch eigene Entwickler, so wird bereits während des Assessments gemeinsam mit den Entwicklern der Source Code analysiert, die Entwickler gecoacht und etwaige schwere Mängel können so ggf. direkt behoben werden.

Auch die Datenmenge wird betrachtet, da die steigende Datenmenge über viele Jahre im Produktiveinsatz die Software an Grenzen bringen kann, die zum Zeitpunkt der Entwicklung nicht antizipierbar waren. Auch hier sind etwaige Tuning und Optimierungsmaßnahmen notwendig, damit die Software nicht überaltert und nicht mehr weiter betrieben werden kann.

Nicht nur die Software, sondern auch die umgebenden Prozesse entscheiden darüber, ob eine Software bedenkenlos weiterbetrieben werden kann. Wie wird beispielsweise bei Änderungen an der Software getestet? Wie hoch ist der Automatisierungsgrad bei Integration und Deployment?



Wie werden Fehler für die Legacy Software, die eingemeldet werden, abgearbeitet?

Im zweiten Teil des Application Quality Quick Assessments werden diese unterschiedlichen Prozessaspekte beleuchtet.

Die Ergebnisse fließen ebenfalls in die Bewertung der Bestandssoftware ein.

Ähnlich einem Pickerlbericht für ein Fahrzeug umfassen die Ergebnisse des SEQIS Application Quality Quick Assessments einen detaillierten Mängel- und Maßnahmenkatalog mit entsprechender Mängeleinstufung, sowie einen schriftlichen Abschlussbericht und eine Abschlusspräsentation.

Sowohl der Softwarecheck als auch der Prozesscheck und die Erstellung der Berichte ist in den üblichen notwendigen 5 Personentagen enthalten.

Gerne bewerten wir auch Ihre Bestandssoftware und Ihren Legacy Code für Sie als fundierte Einschätzung.■

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 25 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.

Interview mit Heinz Wachmann
Geschäftsleitung OeKB Business Services GmbH

Datum: 02. März 2020
Ort: Mödling, Niederösterreich

Heinz Wachmann, Geschäftsleitung OeKB Business Services GmbH, setzte im vergangenen Jahr, bei zwei Projekten, auf das SEQIS Application Quality Quick Assessment.
In maximal fünf Personentagen wurde die Software überprüft und eine Art „Pickerlüberprüfung", als Entscheidungsgrundlage für den weiteren Betrieb, entwickelt.

Helena Thurner, SEQIS: Vielen Dank, dass Sie sich heute Zeit genommen haben, um mit uns dieses Interview zum Topthema Legacy Code zu führen. Inwieweit stellt Legacy Code in Ihrem Unternehmen eine Herausforderung dar?

Heinz Wachmann, OeKB BS:
Legacy Code ist bei uns, wie auch bei vielen anderen Unternehmen, eine Herausforderung, da eine Software nicht von einem Tag auf den anderen entsteht und auch nicht in so kurzer Zeit abgelöst werden kann. Man hat immer irgendwo, wie bei einem Legobausteinsystem, Teile, deren Alter und technische Schuld problematisch sind. Diese Teile kann man nicht so einfach austauschen.

Helena Thurner, SEQIS: In vielen Unternehmen ist es so, dass sie Software im Kundenauftrag erstellen und betreiben. Der Kunde ist jedoch bei individueller Entwicklung trotzdem dafür verantwortlich, etwaige Wartungsmaßnahmen am Legacy Code zu beauftragen und auch von einem kompetenten Partner durchführen zu lassen. Wie ist das in Ihrem Kontext?

Heinz Wachmann, OeKB BS: In manchen Fällen ja, in anderen sind wir aber nur der Software-Wartungspartner. Bei uns gibt es unterschiedliche Modelle. Wir haben Verträge, durch die wir einerseits den Betrieb garantieren - hier gehört die Softwarewartung mit den entsprechenden Haftungen dazu. Da es sich dabei um businesskritisch Applikationen handelt, ist dort das Niveau entsprechend hoch. Andererseits gibt es genauso Software, wo wir nur für die Entwicklung zuständig waren und die Wartung entsprechend durchführen.

Helena Thurner, SEQIS: Sie hatten in den letzten Jahren mehrere Überprüfungen der Legacy Software durch andere Anbieter. In welcher Hinsicht haben Sie die Dienstleistungen von SEQIS und insbesondere das SEQIS Application Quality Quick Assessment positiv überrascht?

Heinz Wachmann, OeKB BS: Wie bereits Goethe gesagt hat: „Ich schreibe Dir einen langen Brief, weil ich keine Zeit habe, einen kurzen zu schreiben". Also inwieweit hat mich die Arbeit positiv überrascht? Wesentlich war und ist, dass der Chef selbst gekocht und serviert hat, und das ist nicht selbstverständlich. Das Zweite ist die Durchgängigkeit von der Schnellanalyse bis hin zur vorstandsfähigen Präsentation. Demnach wurden meine Erwartungen erfüllt.

Helena Thurner, SEQIS GmbH: Wir konnten bei beiden Applikationen bescheinigen, dass diese noch einige Zeit weiter betrieben werden können. Nichtsdestotrotz besteht dringender Handlungsbedarf, insbesondere im Bereich der zugrunde liegenden Laufzeitumgebung (Java) als auch der verwendeten Libraries. Es wäre notwendig, diese laufend zu aktualisieren, um Security-Probleme, die bereits gelöst wurden, auch in der Applikation zu lösen. Woran liegt es Ihrer Meinung nach, dass diese Art der Wartung oft vernachlässigt wurde?

Heinz Wachmann, OeKB BS: Es ist im Prinzip eine Prioritätensetzung. Vergleichen kann man es auch gut mit einem Besuch beim Zahnarzt: Erst dann, wenn einem etwas wirklich Schmerzen bereitet, wird man tatsächlich aktiv. Die Kunden und auch wir selbst haben nicht nur eine Applikation, sondern viele. Insofern ist ein „Pickerl" für eine Software eine angebrachte Bezeichnung, wenn man einmal im Jahr verpflichtet oder angehalten ist, gemeinsam mit einem unabhängigen Experten genau das zu überprüfen. Ansonsten geht das leider im Tagesgeschäft sehr oft unter.

Helena Thurner, SEQIS: Was sollte bei einer Software-„Pickerl-Überprüfung" kontrolliert werden?

Heinz Wachmann, OeKB BS: Alles was wir gemeinsam mit SEQIS beim Softwarecheck und beim Prozesscheck überprüft haben.

Helena Thurner, SEQIS: Ab welchem Alter würden Sie eine regelmäßige Überprüfung der Legacy Software mit dem SEQIS Application Quality Quick Assessment empfehlen?

Heinz Wachmann, OeKB BS: Das hängt sehr stark von den verwendeten Softwarekomponenten ab. Die Applikation, die wir jetzt analysiert haben, ist schon etwas in die Jahre gekommen. Ich würde sagen, dass die Entwicklung in den letzten zehn Jahren exponentiell schneller geworden ist. Wir können die Software, die wir uns jetzt angeschaut haben, nicht auf eine neue Technologie eins zu eins umlegen. Wenn wir zum Beispiel Apps vom Handy betrachten, dann sind diese nach 12 Monaten schon „veraltet". Das bedeutet, es ist sehr stark vom Verwendungszweck und auch von den Möglichkeiten, dass hier Anomalitäten entstehen abhängig. In unserem Fall, bei der Prüfung von Finanz-Applikationen ist ein viel höherer Handlungsbedarf notwendig, als bei einer Spielesoftware. Es kann daher nicht immer vom Alter abhängig gemacht werden, sondern auch vom Einsatzgebiet und der Kritikalität.

Helena Thurner, SEQIS: Welche Maßnahmen haben Sie nun aus den Überprüfungsergebnissen abgeleitet und umgesetzt?

Heinz Wachmann, OeKB BS: Auch hier möchte ich gerne das Beispiel von der TÜV Überprüfung oder vom „Pickerl" anwenden. Wir haben einige Mängel gefunden und deren Behebung sofort in den nächsten Softwarewartungszyklus eingebaut. Die Behebung von Mängel, die sehr zeitintensiv sind, haben wir für die nächsten Releases eingetaktet.

Helena Thurner, SEQIS: Wie kann man dem Kunden transparent vermitteln, dass ihr Unternehmen ein Problem mit der Legacy Software hat und auch laufend in diese investieren muss, damit die Software mit entsprechend geringem Aufwand wartbar bleibt?

Heinz Wachmann, OeKB BS: Das ist wohl ganz individuell zu sehen. Aber gerade wenn die Kostenthematik im Vordergrund steht, macht es aus meiner Perspektive Sinn im Vorhinein eine 5- oder 7-Jahres-Perspektive ins Auge zu fassen. Vielleicht gelingt dann ein anderer Blick auf die Investition.

Helena Thurner, SEQIS: Das heißt, man müsste auch bei einer Inhouse entwickelten Software klar sagen, dass man auch hier, sagen wir ca. 15%, investieren muss, um diese entsprechend zu warten?

Heinz Wachmann, OeKB BS: Zu den Investitionskosten gibt es sicher professionelle Studien. Die Herausforderung ist einfach, dass diese Dinge immer exponentiell mehr werden. Das heißt, am Anfang sind es vielleicht nur 5 % und dann sind es 12 % und dann vielleicht sogar 20 %.

Helena Thurner, SEQIS: Ja, da die Software natürlich auch über die Zeit altert. Schwierig wird es, wenn die Wartung aufgeschoben wird und Fehler korrigiert werden, denn dann werden möglicherweise neue Fehler eingebaut. Die spannende Frage ist natürlich, ob man verhindern kann, dass Software zur Legacy Software wird.

Heinz Wachmann, OeKB BS: In unserem Fall handelt es sich immer um ganz individuelle Softwarelösungen. Die sind nicht auf fünf Jahre ausgelegt, sondern haben einen längeren Lebenszyklus, da sie so speziell sind. Ab dem vierten Jahr sind die technischen Schulden überproportional gestiegen. Wenn Sie eine Standardsoftware haben, dann haben Sie vom Hersteller ca. alle zwei Jahre ein Upgrade. Dieses ist bei einer individuellen Lösung nicht gegeben.

Helena Thurner, SEQIS: Vielen Dank Herr Wachmann, möchten Sie noch etwas ergänzen?

Heinz Wachmann, OeKB BS: Ich möchte noch darauf eingehen, was wir schon alles gemacht haben. Wir haben uns nicht nur den Code angeschaut, sondern auch die Umgebung. Ein wesentlicher Punkt ist die Frage nach der Dokumentation, sodass auch Personen, die nicht direkt an der Softwareentwicklung beteiligt waren, bestens informiert sind und entsprechend weiterarbeiten können. Wesentlich ist es, dass das Wissen unserer Mitarbeitenden auch nach deren Pensionsantritt im Unternehmen erhalten bleibt. Dies ist natürlich mit einem Mehraufwand in der Dokumentationspflege verbunden – jedoch eine Investition mit Mehrwert.

Helena Thurner, SEQIS: Herzlichen Dank für das Interview. Wir freuen uns schon auf viele weitere Projekte, um unsere erfolgreiche Zusammenarbeit fortzusetzen!

Altlast vs New Solution - wann sanieren, wann ablösen?

von Andreas Steiner

Je nach Situation ist es sinnvoller Legacy Code zu erweitern und nach und nach zu überarbeiten („refactoring") oder direkt mit einer Neulösung zu beginnen. In erster Linie ist dazu zu sagen, beides ist ein großes Stück Arbeit an dem man nicht herumkommen wird.

Sehen wir zunächst die testspezifischen Herausforderungen an, die bei der Erweiterung von Legacy Code auf uns zukommen.

Startpunkt und Ausgangsbasis: Wir haben ein Altsystem vor uns.
Aus Testsicht haben wir folgende Rahmenbedingungen:

1. Keine Testfälle. Der Aufwand in dieser Situation auch noch Testfälle für das Bestandssystem zu erstellen, und diese dann durchzuführen, ist oft massiv. Aber der einzige Weg um Überraschungen in Produktion einzuschränken.

2. Gibt es dokumentierte manuelle Blackbox-Testfälle können diese als Regressionstest durchgeführt werden. Allerdings setzen diese aber auch genügend Zeit für Durchführung und Analyse im Fehlerfall voraus. Je nach Anwendungsgebiet und Ressourcen kann dies sinnvoll sein. Analysieren Sie jedenfalls im Vorfeld, ob diese Testfälle auch die Veränderung abdeckt!

3. Gibt es hingegen automatisierte (API-)Tests, können je nach Qualität der Testfälle und Art der Veränderung Fehler rasch identifiziert werden. Die Kehrmedaille: Automatisierte Tests müssen ggf. mit jeder Änderung gewartet werden.

Wichtig wäre jedenfalls für alle 3 Rahmenbedingungen, dass die neu geschaffene Funktionalität auch gleich in Testfälle gegossen wird – zum Erstellungszeitpunkt sind alle Fakten noch frisch und die Ableitung von Testfällen ist eine vergleichsweise einfache und rasche Aufgabe.

Sie sehen: In den meisten Fällen verhärtet sich Legacy Code und ist nur noch mit großem Aufwand qualitätszusichern. Neue Features zu implementieren, Code zu dokumentieren oder auch nur einen kleinen Bug zu fixen benötigt immer mehr Zeit. Das Ergebnis: Software-Entwickler werden zu Software-Lesern, weil sie mehr Code lesen und analysieren müssen als tatsächlich zu entwickeln.

Legacy Code erweitern frisst nicht nur mehr Ressourcen, sondern belastet auch das Team. Der Druck für die Fertigstellung bis zur nächsten Deadline steigt, ebenso der Frust über die Handlungsunfähigkeit. Im Worst Case steigen die Projektkosten ins unermessliche, Aufwände sind nicht mehr akkurat abzuschätzen und die Probleme wirken sich auf das gesamte Unternehmen aus und können sogar zum Ruin führen. Quasi eine Todesspirale...

Doch nicht Altes ändern, sondern doch lieber was Neues dazu stellen...?
Wieso dann überhaupt erweitern bei so einem hohen Risiko? Dann lieber doch gleich ablösen!

Man könnte nun auf eine neue Plattform migrieren mit der Motivation innovationsfähig zu bleiben (das alte System eignet sich für manche Features gar nicht). Weitere Gründe sind u.a.:

- Fachwissen erhalten, welches im Code steckt, aber nicht ausreichend dokumentiert ist

- Teams erhalten, welche ggf. nicht begeistert davon sind, das historisch gewachsene System weiter zu entwickeln

Aber auch hier gibt es viele zu betrachtende Rahmenbedingungen:

- Never change a winning Team. Der alte Code hat sich ja viele Jahre bewährt, aber es ist z.B. aufgrund fehlendes Supports der Schnittstellen nicht möglich neue Features zu implementieren. Migrieren Sie nun die alten Codes auf eine neue Plattform, haben Sie ggf. rascher ein Ergebnis und bleiben konkurrenzfähig, da sie die neusten Features verwenden können. Das sichert Fachwissen und Mitarbeiter! Win-Win! ... oder doch nicht?

- In unserem Beispiel gesellt sich zu Ihrer alten Architektur eine zweite, neuere. Oft wird verabsäumt die neue Zielarchitektur, unter Berücksichtigung der Vor- und Nachteile der beiden Lösungen klar fest zu machen. Jede neuere Entwicklung kann mal da mal dort gemacht werden... und nach ein paar Jahren kommt zur zweiten noch eine dritte Architektur dazu. Quasi: „Neues Feature? -> neue Architektur! Hat ja letztes Mal gut geklappt." Letztlich befindet man sich in einen Teufelskreis. Am Ende steht man vor einem Softwaregebilde, das vermutlich viel schwerer zu warten und viel teurer ist als die ursprüngliche Architektur.

Trotzdem kann ein Parallelsystem zu etablieren eine sinnvolle Option sein. Insbesondere bei kleineren, leistungsstarken und treuen Teams mit gut ausgebildete Coding-Skills - beispielsweise in den Bereichen CMS oder eCommerce – konnte ich das schon mehrfach in der Praxis erleben.

Oder doch lieber: Alles neu!
Befinden Sie sich bereits in der Situation, dass Mitarbeiter gegen ein nicht wartbares Architekturmonster kämpfen und Features zu implementieren einer epischen Schlacht gleicht, ist es wohl höchste Zeit auf ein Neusystem zu setzen.

Es gibt gute Ansätze wie man das Fachwissen in der Bestandssoftware in neue Anforderungen überleitet. Und damit ein neues System zu realisieren, dass die Überlegungen der alten Lösung berücksichtigt.

Same old thinking - same old result. Or Clean Code
Grundbedingung sind ausreichend Schulung & Coaching, wenn Sie Ihr Bestandpersonal mit der neuen Technologie entwickeln lassen wollen. Setzen Sie Standards, wie der neue Quellcode nach den aktuell-gültigen Plattformstandards auszusehen hat - Stichwort Clean Code!

Bei Clean Code geht es aus meiner Sicht nicht darum einfach nur Schulungen zu machen, sondern vielmehr das richtige Mindset zu verinnerlichen. Es gilt ein Gefühl für Ästhetik zu entwickeln, das dazu motivieren soll, „schönen" Code zu schreiben, der lesbar und wartbar ist. Entwickler sehen in ihrem Code häufig nur die Funktionalität, obwohl die Kunst dahinter genauso wichtig ist. Vergleichen Sie es ein bisschen mit Poeten, die Gedichte verfassen. Dies führt nicht nur zu funktionierendem Code, sondern auch zu Erfolgserlebnissen bei den Entwicklern und auch eine Art von Stolz.

Entscheidungshilfe
Folgende Punkte sollten Sie im Rahmen Ihrer Entscheidung „Ändern", „Erweitern" oder „Neu machen" abschätzen:

Quellen und weiterführende Informationen:

(Quelle: https://entwickler.de/online/php/php-legacy-projekte-sanieren-abloesen-299337.html)

https://www.embedded-software-engineering.de/alptraum-legacy-code-wie-profis-damit-umgehen-a-800689/

https://entwickler.de/online/php/php-legacy-projekte-sanieren-abloesen-299337.html

Andreas Steiner, ist Consultant

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.

Die nächsten Termine im Überblick:

28. Mai 2020

Agile Circle Online

April		
1	Mi	
2	Do	
3	Fr	
4	Sa	
5	So	Palmsonntag
6	Mo	
7	Di	
8	Mi	
9	Do	Gründonnerstag
10	Fr	Karfreitag
11	Sa	Karsamstag
12	So	Ostersonntag
13	Mo	Ostermontag
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	

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

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

Agile Circle

Der Treffpunkt für UmsetzerInnen digitaler Innovationen

Die erste Agile Circle ONLINE Konferenz findet am **28. Mai 2020** statt. Denn gerade in diesen Zeiten braucht die Community den Austausch und die Vernetzung. Nehmen Sie daher teil - ganz bequem und sicher von zu Hause aus!

Geplanter Ablauf am 28.05.2020

Am Vormittag starten wir mit **Impulsvorträgen** zur aktuellen agilen Herausforderungen - inkl. **Q&A Sessions.**

Am Nachmittag finden informative **Online Workshops** statt. Danach folgt eine **Podiumsdiskussion** zum Thema „Das Coronavirus und die Auswirkung auf die Arbeitswelt".

Abschließend lassen wir den Tag in einer entspannten Atmosphäre mit einer **Unconference** ausklingen. Hier haben Sie die Möglichkeit noch offene Fragen zu stellen.

Weitere Informationen finden Sie auf unserer Website: www.agilecircle.org

Wir freuen uns auf Ihre Teilnahme!

www.agilecircle.org

Legacy wird abgelöst: Der König ist tot, es lebe der König

von Klemens Loschy

Sie haben also den grundlegenden Entschluss gefasst, ihr Legacy System zu erneuern. Haben Sie ebenfalls schon eine Strategie festgelegt, wie das passieren soll? Sollen Teile zielgerichtet upgegradet werden, um beispielsweise auf aktueller Hardware oder um in einem neuen Environment lauffähig zu werden? Oder geht es eher in die Richtung einer kompletten Ablöse mit neuen Technologien und moderner Architektur? Oder liegt die Wahrheit irgendwo dazwischen? Was auch immer ihr Ziel ist, und wenn es auch noch weit in der Zukunft liegt, auch der Ablöseprozess an sich muss gut durchdacht und vorbereitet werden.

Unabhängige Komplettablöse oder „Step by Step"-Migration?
Diese Entscheidung wird Ihnen manchmal abgenommen: Ist die neue Lösung mit dem abzulösenden System einfach nicht kompatibel, bleibt einem nur die unabhängige Komplettablöse. Das trifft oft bei (customisierten) Off-the-Shelf Lösungen zu. Die Chance, hier von einem OtS auf ein anderes schrittweise zu migrieren, ist gering. Die zumeist stark unterschiedlichen Lösungen eignen sich nicht für eine schrittweise Ablöse.

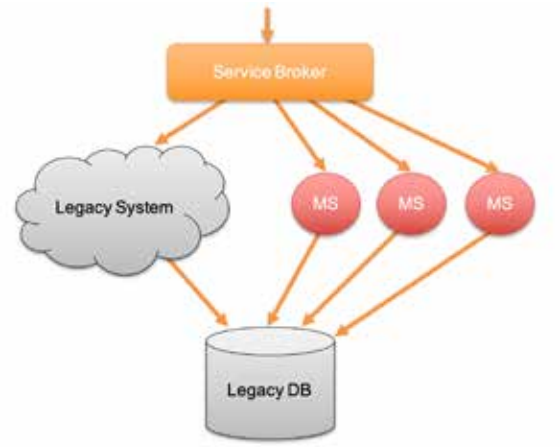
Falls jedoch eine Eigenentwicklung durch eine neue Eigenentwicklung abgelöst werden soll, stellt sich diese Frage sehr wohl. Hier hat man zumindest die theoretische Möglichkeit, den einen oder den anderen Weg zu gehen.

Folgende Faktoren können die Entscheidung beeinflussen:

- Sind die Technologien miteinander kompatibel bzw. kann die neue Umsetzung einfach in die alte integriert werden? Oder schränkt eine Integration die Zielarchitektur nur unnötig ein?
- Wie sehr muss das Legacy System angepasst werden, um eine Integration zu ermöglichen und sind die notwendigen Ressourcen und das notwendige Know-How dafür vorhanden? Können diese Anpassungen einfach getestet werden oder ist das Risiko eines Folgefehlers zu hoch?
- Ist die Integration der beiden Entwicklungen mit hohem zusätzlichem Aufwand verbunden? Gibt es dafür hinreichende Gründe, um den Mehraufwand trotzdem zu rechtfertigen?
- Ist die Integration für den Endbenutzer transparent oder entstehen dadurch nennenswerte Usability-Hürden?

Eine Komplettablöse im Vergleich zu einer „Step by Step"-Migration ist oft einfacher, weil „auf der grünen Wiese" entwickelt werden kann: Technologie, Architektur aber auch die Arbeitsweise und der komplette Entwicklungsprozess können komplett neu und zielgerichtet ausgewählt und umgesetzt werden. Oft nutzt man so ein Projekt auch bewusst als Spielwiese für gänzlich Neues und (zumindest im Unternehmen) bisher Unerprobtes, beispielsweise agile Prozesse oder eine neue Teststrategie. Aber alles mit Bedacht: Schraubt man an zu vielen Stellen gleichzeitig steigen allgemein die Risiken für das Projekt. Ein weiterer Vorteil ist, dass das Legacy System bei einer unabhängigen neuen Lösung komplett unangetastet bleibt - oft zeichnet sich das Legacy System ja gerade dadurch aus, dass es am besten komplett unangetastet bleiben soll (Legacy Code als „code that developers are afraid to change"). Die Komplettablöse hat aber zumindest den Nachteil, dass Erfahrungen mit der neuen Lösung und greifbare Ergebnisse sehr lange (quasi bis zu Fertigstellung und Ablöse) auf sich warten lassen.

Bei der schrittweisen Ablöse werden Teile des Legacy Systems iterativ herausgelöst und durch eine Neuentwicklung ersetzt. Der Grundstein dafür ist mitunter aber arbeitsintensiv, denn das Legacy System ist üblicherweise nicht dafür ausgelegt, auseinandergenommen zu werden. Oft sind dafür zusätzliche Schnittstellen im Legacy System notwendig. Auch der Einsatz eines „Proxies" oder einer Middleware, die die alte und die neue Welt miteinander verbindet, ist nicht unüblich und oft notwendig. Steht diese Architektur aber grundlegend, kann das Legacy System schrittweise abgelöst werden. Dafür eignet sich oft eine auf MicroServices basierende Lösung: das Legacy System wird in funktional zusammenhängende Cluster unterteilt, die jeweils durch ein MicroService abgebildet werden. Die einzelnen MicroServices können direkt, oder bei Bedarf über die neue Middleware, kommunizieren. Die neue Lösung ersetzt also schrittweise das Legacy System, bis alle (notwendigen) Funktionen übernommen wurden und das Legacy System deaktiviert werden kann.



Schrittweise Erweiterung des Legacy Systems durch MicroServices und einem eigenem Service Broker

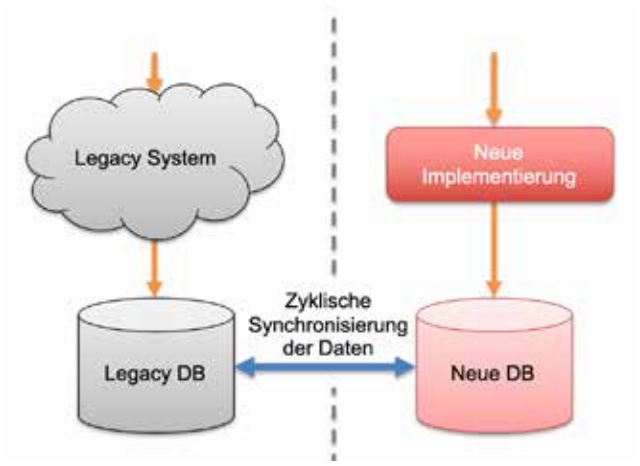
Welche Strategie ist jetzt die bessere? Die Antwort lauten natürlich: it depends! Eine allgemeingültige Aussage dafür gibt es nicht, beide Ansätze haben ihre Vor- und Nachteile, die gegeneinander abgewogen und bewertet werden müssen. Meiner Erfahrung nach ist jedoch eine schrittweise Migration, falls technisch und organisatorisch machbar, oft die bessere Lösung.

Parallelbetrieb oder „Big Bang"?
Egal ob das Legacy System schrittweise migriert oder durch eine neue, unabhängige Lösung ersetzt wird, stellt sich die Frage, wann denn die neue Lösung eingesetzt werden soll? Wartet man die Fertigstellung der neuen Lösung ab oder kann man, bzw. muss man sogar, schon vorher Teile sinnvoll verwenden? Welchen Mehraufwand bedeutet es, beide Lösungen parallel zu betreiben und ist das technisch überhaupt sinnvoll möglich?

Ein Parallelbetrieb verringert das Risiko der eigentlichen Ablöse, denn alle Beteiligten/Betroffenen (Endbenutzer, Entwickler, Tester, Operations, ...) bekommen so die Möglichkeit, frühzeitig und gezielt Erfahrungen mit dem neuen System (oder eben Teilen davon) zu sammeln. Ein Gegensteuern bei ungewollten Ergebnissen ist damit frühzeitig möglich.
Für einen Parallelbetrieb müssen aber die technischen Voraussetzungen stimmen, denn beide Welten müssen im Endeffekt während des Parallelbetriebs voll in das Unternehmen integriert sein. Im Best Case sind beide Systeme so miteinander verbunden, dass zumindest ein Teil der Arbeitsergebnisse in beiden Systemen jederzeit verfügbar ist und so der Endbenutzer die freie Wahl hat, welches System er verwenden möchte. Im Worst Case ist die Datenhaltung strikt getrennt und jeweils nur in einem der beiden Systeme verfügbar. Beide Varianten sind denkbar, wobei der Overhead einer strikt getrennten Datenhaltung meist nur zeitlich stark begrenzt und mit einer kleinen Nutzergruppe Sinn macht.

Die Ablöse durch einen „Big Bang" ist zumindest eines: spannend. Jeder, der schon einmal bei so einem Event dabei war, weiß das. Üblicherweise wird das Legacy System zu einem gewissen Zeitpunkt abgeschaltet und alle Daten in das neue System migriert. Ein Rückstieg auf das Legacy System ist fast niemals geplant oder möglich, da sich der Aufwand dafür nicht rentiert. Diese in Summe wenigen Schritte dauern mitunter mehrere Tage, in denen weder das alte noch das neue System aktiv sind. Teilweise zahlt es sich sogar aus, für diese Downtime eigene Zwischenlösungen zu bauen, wenn der sonst verlorene Umsatz die Entwicklungskosten übersteigt. Ab da an ist nur noch das neue System aktiv, und alle Beteiligten atmen tief durch, wenn der Normalbetrieb wieder erreicht ist. Ein gut durchdachtes, sehr detailliertes und zumindest einmal erprobtes Umstiegsszenario ist dabei jedenfalls Pflicht und hilft in diesen angespannten Stunden oder Tagen immens!

Auch diesmal stellt sich die Frage, welches Szenario denn das bessere ist, und Sie ahnen es wohl bereits: it depends (again)! Der Parallelbetrieb ist initial aufwendiger, wenn er überhaupt möglich ist. Aber dann ist er dem „Big Bang" meiner Erfahrung nach vorzuziehen. Die Erfahrungen



Parallelbetrieb der beiden unabhängigen Systemen (Legacy und Neu) mit zyklischer Synchronisierung der Daten

aus dem Parallelbetrieb und das nicht vorhandene „Umstiegswochenende" überwiegen hier. Aber auch Aspekte, die nicht technischer Natur sind, haben Einfluss auf die Entscheidung: Es kann sein, dass die neue Lösung so früh wie möglich platziert werden soll, auch wenn der Funktionsumfang noch weit hinter dem des Legacy Systems liegt und der Aufwand für den Parallelbetrieb enorm ist. Doch gerade, wenn es darum geht, dass ein Unternehmen auf dem Markt sich einen Vorteil beim Mitbewerb dadurch erhofft, wird diese Strategie gerne verfolgt.

Die Ablöse von Legacy Systemen ist niemals einfach und die Strategie zur Ablöse muss gezielt auf die jeweilige Situation angepasst werden. Und dennoch ist so eine Ablöse an sich ein normaler Prozess im Software LifeCycle und hat den (eher) schlechten Ruf wahrlich nicht verdient. Ablöseprojekte sind spannend und anspruchsvoll und eine Chance, viele Dinge besser zu machen und Neues auszuprobieren. Nutzen wir doch diese Chance und werden wir unsere Altlasten los, besser früher noch als später!■

Titel: „Farbe", Künstler: Edona Zeka, Technik: Acryl-Bild

Klemens Loschy ist Principal Consultant, Teamlead bei SEQIS.

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

Systemwechsel: Wie bringen wir Menschen dazu sich auf das Neue zu freuen und das Bewährte zurückzulassen

von Alexander Weichselberger

Wer sich mit der Thematik Legacy Code auseinandersetzt, kommt meines Erachtens im technischen Bereich nicht um „Working Effectively with Legacy Code“ von Michael C. Feathers herum. Aber sehr schnell gehen die Analysen in die Gegenüberstellung von Änderung, Erweiterung bis hin zu einem kompletten Systemwechsel.

Und sollten Sie sich für einen Systemwechsel entschieden haben, starten Sie in diesen Wechsel jedenfalls mit einer gehörigen Portion Change-Management. Denn eines ist von Anfang an klar: Da kommt was komplett Neues. Der Fokus in diesem Artikel liegt in den Routinen, die es braucht, Menschen dazu zu bringen, sich auf Neues zu freuen und das Bewährte zurückzulassen.

Die Herausforderungen

Ganz klar – durch Einführung einer neuen Softwarelösung wird sich die Arbeitsweise im Unternehmen stark verändern. Neue Software, die man lernen muss, neue Prozesse und Abläufe, die man einführen muss, neue Funktionen, die man nutzen muss, ... Im Regelfall werden viele bewährte Routinen, Funktionen und Prozesse, die bislang das daily business bestimmt haben, verändert. Und seien wir uns ehrlich, der klassische Spannungsbogen aller Projekte zwischen *Timeline* <> *Budget* <> *Quality* schlägt wahrscheinlich wieder zu und allzu oft ist die „erste Release“ der neuen Software funktional reduziert. Es kommt weniger, wie initial geplant. Aus Sicht der Endanwender (EA) zusammengefasst:

- Mehrleistungen und viel Arbeit für die Entwicklung des neuen Systems, wie zum Beispiel Analyse und Anforderungsworkshops, (Teil)Projektleitertätigkeiten und/oder ProductOwner-Aktivitäten, usw. – parallel zum daily business im Unternehmen
- Was es funktional schon in der „Altlösung“ gegeben hat und für das vermeintliche Überleben im Tagesgeschäft notwendig ist, kommt nicht zum Go Live, sondern erst in einer späteren Release
- Ausbildung, Training, Testunterstützung, Dokumentation: Was angekündigt wurde, hält nicht bzw. muss kurzfristig oft zumindest reduziert werden
- À contre will die Unternehmensleitung, dass die neuen Funktionen auch gleich mehr Ergebnis generieren – im Regelfall gleich in neuen Geschäftsbereichen, die mit dem Go Live möglich sind

Darüber hinaus ist auch die Art und Weise des Go Live aus EA-Sicht relevant: Big Bang, mit dem großen Umstiegsrisiko (EA: „Na, ob das gut geht!“) vs. schrittweiser Einführung (EA: „Da kommt immer wieder was Neues – wir können lange nicht Prozesse und Vorgehen stabilisieren“) vs Parallelbetrieb der alten und neuen Lösung (EA: „Jetzt muss ich beides machen!“) – beim Fachbereich generiert das jedenfalls jede Menge Arbeit!

Durch ein dediziertes Change Management (CM) sollen Probleme rasch identifiziert und einer Lösung zugeführt werden. Proaktives Abfragen und intensive Kommunikation mit den Stakeholdern sollen den Umstieg erleichtern.

Spezialfall „Einführung einer Standardsoftware“

Falls Sie eine Standardsoftware a la Microsoft Dynamics oder SAP einführen, gilt es auch die entsprechende Analyse der eigenen Anforderungen generell am Konzept „Wir passen uns an die Standardsoftware an“ zu orientieren. Durch Konfiguration der Software („Customizing“) passen Sie die Software grundsätzlich an Ihre Unternehmen an. Wenn Sie darüber hinaus auch noch (viele) Individualentwicklungen fordern, laufen Sie in Gefahr, zu weit aus dem Standard zu laufen – Sie entwickeln eine Individualsoftware mit der Ausgangsbasis Standardsoftware. Zu schnell baut man auch systemfremde Funktionen ein und macht aus z.B. einer ERP ein „ERP+CRM+BI+...“ . Halten Sie das System sauber! Denken Sie daher eher daran, Zusatzfunktionen in vor- oder nachgelagerten Systemen auszulagern, wie z.B. Datawarehouses und kümmern Sie sich um eine ordentliche Datendrehscheibe (Middleware) zwischen Ihren Systemen.

Die Anforderungen an die Endanwender-Vertreter

Wenn wir die Anforderungen an das neue System zusammenstellen, so hat der EA bzw. deren Vertretung im Projekt eine gewichtige Rolle: Systeme werden schließlich (im Regelfall) nicht für die ausführende IT gebaut, sondern dienen vorrangig dem EA. Somit brauchen wir Anforderungen und schließlich auch deren Acceptance durch den EA.

Aufgaben, die sehr oft auch von den

Vertretern der EA zu begleiten sind: Workshops zur Prozessanpassung, Training der eigenen KollegInnen, fachliche Dokumentation der neuen Lösung, Test und die zugehörigen Abstimmungen im Defect Boards, Teilnahme und Entscheidungsaufbereitung für das Change Advisory Board (CAB),... damit wird augenscheinlich, wer einen großen Teil der Last des Einführungsprojekts tragen muss.

Projektablauf – Änderungen gemanagt

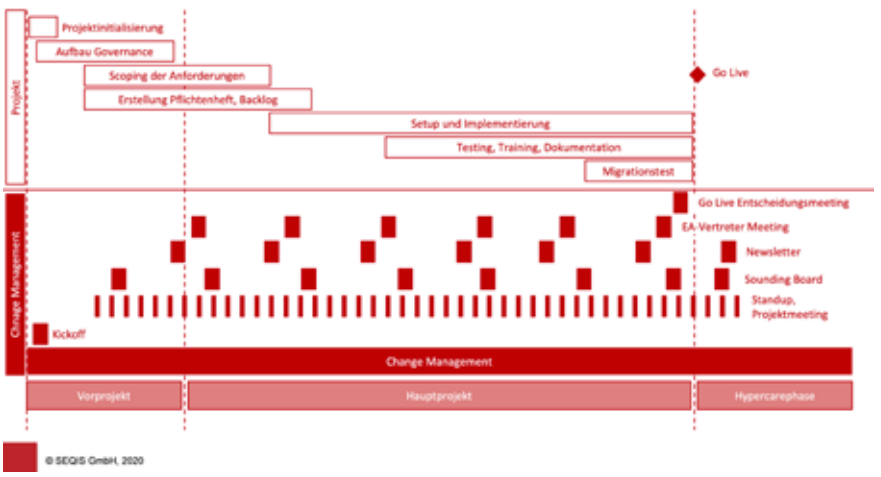


Abbildung 1 - Change Management in Standard Projektsituationen

Wie in obenstehender Abbildung dargestellt stehen spezifische Change Management Aktivitäten Projektstandards gegenüber.[1]

KickOff

Am Projektstart ist sicherlich eine Art ***KickOff*** geplant. Dabei ist es übliche Aufgabe des Change-Managements, neben den eigenen Aktivitäten im Projekt, die Projektziele und -gründe aus fachlicher Sicht vorzustellen:

- Warum ist das Projekt zu machen?
- Was passiert, wenn das Projekt nicht wie geplant realisiert werden kann oder kann nichts gemacht wird?
- Wie ist die fachliche / IT seitige Abgrenzung von Arbeiten vorgesehen?

Darüber hinaus sind auch die Projektspielregeln vorzustellen. Dies hat den Vorteil, dass die prozessorale Dimension auch des Projekts beim Change Management liegt – denn auch diese neuen projektspezifischen Prozesse müssen in der Organisation eingeführt werden. Es ist auch zu erwarten, dass diese Spielregeln über die Zeit noch detaillierter, erweitert oder geändert werden. Wieder Change... . Oft steht auch bereits hier der Ansatz, wie die Zusammenarbeit im Projekt vorgesehen ist (Stichwort: Toolchain) fest und sollten kurz thematisiert werden, gerne auch vom Change Management. Die Überlegung ist: Wenn das Change Management Projektprozesse versteht und einführen kann, dass ist das auch für die Fachlichkeit des Unternehmens möglich. Die EA lernen Zugänge und Argumentationen kennen und können sich noch an Punkten der Projektorganisation „reiben", die im Vergleich zu den Fachprozessen wenig Gewicht haben - besser bei WIE DAS PROJEKT MACHEN, wie beim fachlichen Inhalt des Projekts.

Abhängig von der Dauer des KickOffs könnte Change Management auch Aktivatoren in das Meeting einbauen: z. B. haben Vorstellungsrunden innerhalb der einzelnen Subteams eine lockernde, verbindliche Wirkung, wenn sie spielerlisch instrumentaliziert werden:

- Selbstvorstellung auf Teamebene (dazu stellt man sich ggf. im Kreis auf und wirft den jeweiligen Sprecher einen Ball zu); z. B. „Mein Name ist weixi und meine drei Hashtags sind #Gitarre, #Familie und #Digitalisierung." – eben kurz, knapp und knackig
- Vorstellung eines anderen Teammitglieds – dauert etwas länger und läuft in zwei Runden: Runde 1 – sich bei einem anderen Teammitglied vorstellen und sie/ihn auch kennen zu lernen; Runde 2 – Jeder stellt seinen Interviewpartner in der gesamt Subteamrunde vor

Erweiternd dazu hat es sich bewährt, wenn das Projekt Management im Rahmen des KickOffs die konkreten Projektpläne und -phasen vorstellt, die Teams im Detail (inkl. Funktionen und fachliche Inhalte), die Meetingstruktur (welche Meetings wann, Dauer & Ziele) und vorgesehene (geschätzte) Aufwände für die einzelnen Teams bzw. Projektmitarbeiter sowie den Zwischenstand zu den bisherigen Entscheidungen präsentiert.

Planen Sie jedenfalls ausreichend Zeit für Q&A Sessions mit den Projektmitgliedern ein!

StandUp & Projektmeetings

Machen Sie als Change ManagerIn bei den Standardmeetings mit; insbesondere dort, wo fachlichen Fragen besprochen und entschieden werden. Als Change Manager muss man über Änderungen Bescheid wissen – und sich überlegen, wie man diese

[1] Vereinfachend wurden die unterschiedlichen Projektmanagementmethoden im Wesentlichen nicht unterschieden; diese Darstellung dient lediglich der Verdeutlichung von Intervall, Inhalt und Phasenbezug; selbstredend ist auch diese Aufbau- und Ablaufplan im Rahmen der Governance des Projekts abzustimmen

Änderung in den EA Kreisen verankert.

Sehr häufig kommt den (daily) ***StandUps*** eine durchaus steuernde Wirkung in den Projekten zu. Sie kennen die drei Kernfragen: Was habe ich seit dem letzten StandUp erledigt? Was mache ich bis zum nächsten StandUp? Wo liegt Schwierigkeiten, wo brauche ich Unterstützung? Nehmen Sie teil – damit kennen Sie aktuelle Lage und insbesondere auch Punkte, die gerade schwieriger zu nehmen sind. Vielleicht sind das ja gerade Herausforderungen, die Sie gut aus dem Weg räumen können – und damit Zufriedenheit und Zustimmung im Projekt verbessern.

Neben den StandUps sind auch in den anderen ***Projektmeetings*** viele Ansatzpunkte für Change Management gegeben: Planungs- und Schätzmeetings, technische Abstimmrunden, Abnahme- und Defectmeetings – machen Sie mit, nehmen Sie „Ihre" Standardfragen mit: „Wo ergeben sich prozessorale Lücken?", „Ist das fachlich rund?", „Wie neu ist das für uns im Hause?"

Darüber hinaus: Sollten fachliche Änderungswünsche abgestimmt werden (z. B. im Change Advisory Board oder bei CR Meeting): Diese Meetings haben nicht umsonst „Change" im Namen und „gehören" eigentlich dem / der Change ManagerIn.

Wann richtet man ein Sounding Board ein?	Allgemein gesprochen immer dann, wenn Änderungen im Gange sind oder größere Projekte am Laufen sind.
Warum?	Es unterstützt dabei, Feedback von allen Stakeholdern zu bekommen.
Wer nimmt teil?	Alle Stakeholder des Projekts, dh. sehr oft: ▪ Projektleiter, Keyuser bzw. Teilprojektleiter & Change Management ▪ Betroffene Mitarbeiter / Führungskräfte ▪ Externe Experten ▪ Auftraggeber / C-Level ▪ Betriebsrat
Wie kommt man zu Teilnehmern beim Sounding Board?	Während die offiziellen" Vertreter zu bestimmen eine vergleichsweise leichte Übung ist (Blick auf Projektstruktur bzw. Org-Chart) ist es deutlich schwieriger, die tlw. informellen Meinungsmacher und Führungspersönlichkeiten im Unternehmen zu erreichen. Hier empfiehlt sich eine Promotion (Plakate, Info an die Mitarbeiter, Info durch die Heads > MA >...), bei der man auf die Wichtigkeit hinweist und die Player ins Sounding Board einlädt. Wichtig dabei ist jedenfalls, dass jeder Bereich vertreten ist. Im Zusammenhang kann auch ein Wechsel von Teilnehmern über die Zeit vorkommen und sollte auch unterstützt werden - nicht immer ist eine spezifische Person auch wirklich betroffen und kann daher über die Zeit auch „nicht der / die Richtige" sein. Der Organisator des Sounding Boards sollte jedoch sicherstellen, dass Änderungen transparent sind und eben auch alle Bereiche vertreten sind.
Was passiert beim Sounding Board?	Die Teilnehmer besprechen Status und aktuelle Wahrnehmungen zu den Veränderungen. Im Zusammenhang empfiehlt sich der Ablauf auf Basis ->Lean Coffee. Die im Sounding Board besprochenen Punkte sind zu dokumentieren und die abgeleiteten Aktivitäten und Maßnahmen umzusetzen.
Wie oft, wie lange?	Bei intensiven und umfangreichen Änderun-gen sollte das Sounding Board, wenn es in Form eines Meetings abgehalten wird, 1x pro Monat stattfinden und kann bis zu 90 Minuten dauern.

Abbildung 2 - Fragen und Antworten zum Soundingboard

Sounding Board
Das „Sounding Board" (engl: der Resonanzboden) ist ein Meeting Setup, der während Veränderungen im Unternehmen oder bei Projekten helfen soll, Feedback von den Stakeholdern abzuholen und den aktuellen Status zu bestimmen.
Durch das Soundingboard werden eine Vielzahl von Vorteilen realisiert: Sie erreichen damit:

- Feedback von Stakeholdern
- Verbesserung der Akzeptanz
- Reflektionsmöglichkeit auf Meinungen, Stimmungstrends und „Vibrations" im Unterehmen
- Ermöglicht frühzeitige Kurskorrekturen
- Baut Vertrauen auf

Holen Sie sich diese Vorteile für Ihr Projekt!

Lean Coffee®

Lean Coffee ist ein strukturiertes Meeting, das ohne Agenda startet. Gleich zu Beginn sammeln und priorisieren die Teilnehmer die Agenda gemeinsam. Damit ist sichergestellt, dass für alle Teilnehmer „was interessantes" besprochen wird.
Der Moderator stellt sicher, dass der Ablauf strikt eingehalten wird, insbesondere auch auf die Besprechungszyklen.
Besprochen wird, was durch die Teilnehmen ins Meeting mitgebracht wurden und im Backlog priosiert wurde. Weitere Details siehe http://leancoffee.org/.

Sidestep Follow-Up

Bei vielen Meetings ist das Follow-Up die eigentliche Herausforderung! „Ein nettes Gespräch" ohne Erledigung der ToDo's oder ohne dem strukturiertem Festhalten von Entscheidungen führt oft dazu, dass es lediglich bei dem „netten Gespräch" bleibt, erledigt wird jedoch nichts! Somit empfiehlt hielt sich das Follow-Up abzusichern.
Eine praxistaugliche Übung ist, dass der Teilnehmer, der das Thema eingebracht hat, die Entscheidung und ToDo's in einem vorher vereinbarten Protokoll / Tool abbildet. Damit verteilt man den Aufwand für das das Protokollieren & der Themeneinbringer hat selbst ein vitales Interesse „sein" Thema ordentlich abzubilden.

Kommunikation
Neben der Teilnahme an Meetings und Soundingboards haben sich in der Praxis auch indirekte Kommunikationsvarianten empfohlen: Projektnewsletter oder -mailings, die die aktuelle Situation beschreiben und allen, die nicht unmittelbar im Projekt beteiligt sind, auch zu informieren.

Stellen Sie jedenfalls sicher, wenn Sie solche Newsletter verfassen, dass Sie diese locker gestalten und man „gerne" diese Informationen liest. Fotos von Veranstaltungen, Screenshots, ... sind obligat: Ein Bild sagt mehr wie tausend Worte!

Zyklische Endanwender-Vertreter Meeting
Reden Sie mit den EA-Vertretern; zyklisch, ohne Agenda – hören Sie in Ihr Vis-a-vis hinein und versuchen Sie Probleme in den jeweiligen Bereichen zu identifizieren. Oft versuchen die EA-Vertreter die Last allein zu tragen und gehen damit oft erst viel zu spät „public".

Im Laufe des Gesprächs empfehlen sich: Offene Fragestellungen („Wie beurteilst du...?") und Bestätigungen („Genau, das kann ich mir lebhaft vorstellen!". Regen Sie Reflektionen an („Wie war für dich...?") und fassen Sie in eigenen Worten zusammen – damit erhöhen Sie die Wahrscheinlichkeit auch wirklich alles verstanden zu haben. Ich würde hier weniger im Detail dokumentieren – abgesehen von den vereinbarten ToDos, natürlich.

Treffen Sie sich zu one-on-one's, ist nichts zu verbessern, dann ist mit dieser 1 Stunde nicht viel verhakt. Gibt es aber ein Problem, dass Sie gleich adressieren können: Dann ist diese Stunde sicher Gold wert und schafft Ihnen wertvollen Zeitvorteil. Nutzen Sie dann Ihr Netzwerk und Ihre Projektkompetenzen für die Sache.

Go Live Entscheidungsmeeting
... sicher eine Veranstaltung der Projektleitung.

Aber nehmen Sie aber auch dort Ihre Change Management Rolle wahr, machen Sie klar, was sich nach dem Go Live für den Fachbereich ändern wird: Neuigkeiten in der Software und den Prozessen, aber unbedingt auch Drawbacks, wie offene Bugs, Schulungsstatus, Personalbereitstellung im Kontext zu den antizipierten Problemen, usw. . Sie sollten letztlich in der Lage sein, den Go Live zu empfehlen – oder ggf. eben auch noch nicht. Durch eine sachliche Darstellung unterstreichen Sie die Fürsprecherrolle für den EA, die Sie im Laufe des Projekts aufgebaut haben.

Zusammengefasst
Change Management bedeutet im Kern, dass man Änderungen proaktiv managt. Helfen Sie dem Projekt, indem Sie die EA dabei zu unterstützen, besser vorbereitet, verstanden und unterstützt im Projekt zu sein. Und dabei die Projektziele tatsächlich zu realisieren. ■

Quellen/Referenzen

Working Effectively with Legacy Code, Michael C. Feathers aus der Robert C. Marting Series

Abkürzungen

- BI – Business Intelligence
- CAB – Change Advisory Board
- CM – Change Management
- CR – Change Request
- EA – EndanwenderIn
- ERP – Enterprise Resource Planning
- CRM – Customer Relationship Management

Titel: „Der verlassene Ort", Gruppenbild Malgruppe, Technik: Fließtechnik Fluid Acrylfarben

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.

„SEQIS hat bei der kurzfristigen Übernahme der temporären Testmanagementposition enorm viel Flexibilität und Professionalität bewiesen. Durch die eingebrachte umfangreiche Erfahrung konnte die Position in einer sehr kritischen Projektphase rasch und vollinhaltlich zur Zufriedenheit des Endkunden ausgefüllt werden. Hansjörg hat sich mit seiner umsichtigen und vorausschauenden Art perfekt ins Projektteam eingefügt."

Gerhard Weissinger, Change Program Manager / Head of Customer Team

Die Aufgabe

- *Kunde:* inet-logistics
- *Arbeitsauftrag:* Übernahme des Testmanagements für ein laufendes Kundenprojekt
- *Umfang:* ca. 1.000 Stunden in der Zeit von Oktober 2018 bis Juni 2019
- *Tools:* Jira, Confluence, Zephyr

Die Lösung

- Erfolgreiche kurzfristige Übernahme des Testmanagements
- Testplanung und Steuerung sowie Testdurchführung für drei Releases bis zur Fertigstellung des Projektes
- Durchführung eines UAT (User Acceptance Test) bei einem führenden deutschen Premiumautohersteller zusammen mit einem Consultant der inet-logistics
- Abstimmung, Schulung und Einführung von „Session Based Testing" für die weiteren UATs.

inet-logistics – Schnelle Hilfe war gefragt

Testmanagement und Testunterstützung für ein aktuelles Kundenprojekt

Personalfluktuation kann Unternehmen exponieren. Durch die Schwierigkeit, kurzfristig am Markt eine entsprechende Personalaufnahme durchzuführen, entstand bei inet Bedarf im Software Test. SEQIS konnte rasch und unbürokratisch einspringen und die Lücke erfolgreich schließen. inet, einer der führenden Hersteller von Transportmanagementsystemen, adaptiert und entwickelt Standardlösungen u.a. im Bereich Behältermanagement für Big Player wie z.B. einen führenden deutschen Premiumautohersteller. Projekte in diesem Bereich laufen üblicherweise über Jahre, rd. 9 Monate vor Projektende fiel kurzfristig eine Schlüsselressource im Testingteam aus. SEQIS konnte mit Hansjörg Münster einen erfahrenen Testmanager einphasen, der sehr kurzfristig Agenden übernehmen und das Projekt erfolgreich zu Ende führen konnte.

inet
part of alpega

Mit dem cloud-basierten Transportmanagementsystem (TMS) von inet können Sie Sendungen planen, Ressourcen verwalten, Prozesse optimieren und automatisieren, Transportkosten kontrollieren und Bewegungen in Echtzeit über die gesamte Supply Chain hinweg verfolgen. Es liefert zudem Analytics-Funktionen für bessere Entscheidungen und kontinuierliche Verbesserungen.

Behältermanagement – Müllvermeidung und Kosteneinsparung

Konkret geht es bei der Gesamtlösung „Behältermanagement", neben klassischen Containern und Paletten, auch um hochpreisige Spezialverpackungen für z.B. Motoren und Getriebe. Mit Hilfe des inet Behältermanagement optimiert der Automobilhersteller heute den für den Transport notwendigen Leergutbedarf:

- Aktive Lagerbestandsführung entlang der gesamten Wertschöpfungskette inkl. aller Zulieferbetriebe und Frachtführer
- Optimierte Planung und Beschaffung notwendiger Container durch spezielle Prognoseverfahren
- Bestandsübersicht und Kontrolle über Verbrauch und Schwund auf Knopfdruck durch integrierte Inventurfunktionen

Mit „Behältermanagement" spart inet ihren Kunden Millionen.

Challenge taken

Zum Einstiegszeitpunkt war das Projekt bereits mehrere Jahre im Laufen, viele Herausforderungen konnten genommen werden und auch organisatorisch hatte das Projekt schon viele Hürden gemeistert. Aber es galt vor diesem Hintergrund den engen Zeitplan und die Beziehung zum Kunden in Balance zu halten.

D.h. die Ausgangslage für SEQIS war nahezu klassisch: In kurzer Zeit Aufgaben übernehmen, sich in Prozesse und Arbeitsweisen des Kunden eindenken und „nebenbei" jahrelange aufgebautes fachliches Knowhow lernen. Schnell, umfassend und so, dass es für alle anderen keine negativen Auswirkungen auf deren Ergebnisdruck hat und das Projekt nicht gefährdet wird.

Durch praxisbasiertes Testing Knowhow und SEQIS Standards für den KickDown zum Projektbeginn konnte die Lage rasch erfasst und das fachliche Knowhow schnell etabliert werden. „Wir bei SEQIS machen Projekte erfolgreich. Die notwendige Verlegung des Arbeitsplatzes für 6 Wochen nach Dornbirn, um die Übernahme zu optimieren, war für uns schlicht selbstverständlich." reflektiert Hansjörg Münster, der durchführende SEQIS Testmanager, den Start ins inet Projekt.

„Am Ende dieser Zeit stand noch die gemeinsame Testdurchführung und Regression des damals anstehenden Releases. Dies konnte auch durch die kollegiale Zusammenarbeit im Team erfolgreich gemeistert werden – der vor Ort Einsatz hat sich wieder bewährt."

inet setzt in ihrer Testing Toolchain stark auf Testautomation und verwendet dafür BDD (= Behaviour Driven Design). Die Umsetzung erfolgt mittels des Cucumber–Frameworks und die Testfälle, ausformuliert im Gherkin-Style, werden im Gitlab verwaltet und durch Bamboo durchgeführt. Als weitere zentrale Tools sind die bekannten Atlassian-Tools Jira und Confluence im Einsatz, wobei Jira um das Testmanagementtool ZEPHYR erweitert wurde.

„Eine weitere Rahmenbedingung im Projekt war die weltweite Verteilung der Entwicklungsressourcen. Das Headoffice befindet sich in Vorarlberg, eine Entwicklungsabteilung ist in Wien und eine weitere in Thailand. Durch die aktuellen Kommunikationsmöglichkeiten ist diese weltweite Remote Collaboration bereits Alltag und funktioniert sehr ordentlich." so Hansjörg Münster über die allgemeinen Voraussetzungen im Projekt.

Neben Planung, Steuerung und Durchführung der Testaktivitäten, die in der agilen SW-Entwicklung Alltag sind, war auch der UAT (User Acceptance Test) vor Ort bei einem deutschen Premiumautohersteller eine Aufgabe für den SEQIS Testmanager. „Die Abnahmen durch den Kunden konnten wir im Team mit den inet Kollegen erfolgreich begleiten; natürlich waren diese Show-Downs auch immer recht spannend." resümiert Hansjörg Münster. In diesem Zusammenhang wurde – zur Vereinfachung der fachlichen Abnahmen durch den Kunden – durch SEQIS „Session Based Testing" mit dem Automobilhersteller abgestimmt, geschult und eingeführt. Bis zum Projektende wurden durch den SEQIS Consultant auch weitere Releases tatkräftig unterstützt und konnten erfolgreich zum Produktionseinsatz gebracht werden. Nach Beendigung des Projektes mit dem letzten Release und der nachträgliche Stabilisierungsphase endete für Hansjörg Münster der Einsatz bei inet.

inet
part of alpega

Schnelle Hilfe: Der Kunde inet-logistics

inet ist ein führender Anbieter cloud-basierter Transportmanagementsysteme (TMS). Die Software verbindet globale und multimodale Transportnetze im In- und Outbound. Transportkosten werden um ca. 20% reduziert, indem alle Supply Chain Partner auf einer Plattform online vernetzt werden.

Seit 2018 gehört inet zur Alpega Gruppe, die Komplettlösungen für alle Transportbedürfnisse anbietet, darunter TMS und Frachtenbörsen.

www.alpegagroup.com

BDD & Gherkin

BDD (Behavior Driven Development - auf Deutsch auch verhaltensgetriebene Softwareentwicklung) ist eine in der agilen SW-Entwicklung häufig angewandte Technik um bereits die Anforderungsanalyse so zu formulieren, dass diese Texte zu automatisiert ausführbaren Testfällen werden können.

Die dabei verwendete Sprache ist in ein Regelschema oder Format gepresst. Damit lassen sich im Laufe der Entwicklung diese Anforderungen relativ einfach automatisiert absichern. Eines dieser Formate ist „Gherkin". Die Schlüsselwörter bei „Gherkin" sind: Given, When, Then, And, ...

Ein Beispiel:
Given: A customer orders an article in the Webshop
And: there are 10 articles on stock
When: a delivery order is send to the warehouse
Then: only 9 articles are on stock
And: a confirmation mail is sent to the customer.

Ein Framework zur Umsetzung von BDD ist Cucumber. Ursprünglich war Cucumber für die Programmier-sprache Ruby entwickelt. Heute wird eine Reihe weiterer Sprachen wie Java, Javascript und C++ unterstützt.

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.

Remote Meeting - wann dann, wenn nicht jetzt!?

von Alexander Weichselberger

Sicher, in Zeiten von Corona und Quarantäne ist es obligat, sich abzustimmen, ohne sich physisch nahe zu sein. Und ja, Meetings mit 1-2 m Abstand voneinander, ohne Körperkontakt und Fokus auf „Distanz" funktionieren – aber die jeweilige Anreise exponiert doch alle TeilnehmerInnen bzw. schränken Unternehmen diese Reisen aktuell ein („travel ban").

Aber darüber hinaus gab es Ende 2019 mit der Bewegung um Greta Thunberg auch genug weitere Gründe, sich nicht zu treffen – alle mit dem wesentlichen Ziel die Umwelt zu schonen, Stichwort: Reduktion CO2 Ausstoß. Weitere Gründe für Remote Meetings? Overhead durch Anreisen reduzieren, global work ermöglichen, wenn notwendige personelle Ressourcen nicht vor Ort verfügbar sind, usw. .

Zusammengefasst: Es gibt viele gute Gründe, warum Remote Work notwendig ist. Folgend Empfehlungen aus der Praxis, wenn es um Remote Work, im Speziellen um Remote Meetings, geht.

Wir bei SEQIS – als Anbieter in der IT und Digitalisierungsbranche – setzen seit Jahren in unseren Projekten auf die Möglichkeiten remote zu arbeiten, insbesondere auch darauf Meetings in Form vom Videokonferenzen abzuhalten. Im Zusammenhang haben sich folgende Empfehlung und Spielregeln bei uns etabliert:

Reaktionsfähig und erreichbar sein

Es ist gut, wenn man während der Arbeitszeit, die man sich selbst eingeteilt hat, erreichbar ist. Dh. Telefon in unmittelbare Nähe, Chats & Email aktiv, usw.
Für Meetings bzw. für Zeiten, wo man ungestört und konzentriert arbeiten muss, sollte man diese „interrupts" allerdings deaktivieren. Das bedeutet auch z. B. das Handy „umzudrehen" (Display nach unten) oder ganz wegzulegen (außer, das Telefon selbst wird für das Meeting gebraucht). Untersuchungen zu folge schauen wir tgl. rd. 100 mal aufs Handy, d.h. ausreichend Potential für unnütze Ablenkung... .

Zeiten für Remote Work
... hier sind Kernzeiten für die wechselseitige (spontane) Erreichbarkeit wichtig. Gleichen Sie diese Zeiten ab - idealerweise finden Sie auch Kernzeiten, wo Meetings einfach zu machen sind (Stichwort: globale Verteilung der Arbeitszeiten).

Natürlich bedingt dies auch, dass die technische Infrastruktur gegeben ist. Probleme oder Herausforderungen sollten mit dem jeweiligen technischen Support des Unternehmens abgestimmt werden.

Erlaubte Orte für Remote Work

Definieren Sie auch bitte „erlaubte Orte" für Remote Work - irgendein Café um die Ecke oder dgl. bietet ggf. nicht ausreichend Sicherheit - sei es Blickschutz (hier könnten Blickschutzfilter am Laptop helfen), aber auch Netzwerkinfrastruktur. Es gilt, Unternehmens- und Kundendaten unter allen Umständen zu sichern!

Arbeitsplatz einrichten, Routinen fixieren

... ein separates Arbeitszimmer ist natürlich ein Hit - aber nicht immer gegeben. Wenn Sie keinen Extra-Raum dafür haben, kann man z. B. den Esstisch zum Schreibtisch umfunktionieren. Achten Sie jedenfalls auf:

- alles Unnötige vom Tisch entfernen
- Arbeitsplatzergonomie
  - alles 90-Grad: Oberschenkel zu Unterschenkel, Ober- zu Unterarme
  - Handflächen und Ellenbogen sind auf einer Ebene mit Tastatur und Maus
  - Füße: Fest am Boden, ggf. nutzen Sie auch einen Fusshocker
  - Entfernung vom Monitor: mind. 50 cm, nicht näher
- Check: Passt die Lichtqualität und auch die Arbeitshöhe; auf welchem Sessel sitzen Sie - passt das? Ggf. kann man sich auch ergonomische Sitzkissen für zuhause organisieren
- Routinen festlegen:
  - Jeden Tag zur gleichen Zeit anfangen (geplant, nicht trödeln und noch hier und da was anderes machen), und damit auch pünktlich aufhören
  - Kaffee und co schmecken auch am Schreibtisch recht gut
  - Planen Sie auch die Mittagspause fest ein - es ist auch zuhause notwendig, sich auszuruhen und wieder Energiereserven aufzufüllen (essen, Bildschirmpause, usw.)
  - ... am Arbeitsende ist Schluss mit der Arbeit: Richtig Abschalten dadurch unterstützen, dass der Esstisch wieder frei wird, oder zumindest der Monitor ausgeschalten und die Unterlagen auf einen Stapel gesammelt werden. In der Freizeit sollte uns nichts an die Arbeit erinnern
- Kaffeepause mit Kollegen machen - per Videokonferenz auch mal „die Tassen hoch" und sozial interagieren („Wie geht's?", „Hast du gehört, dass ...", usw.)
- Bewegte Pausen machen: ... Bewegung ist im Homeoffice ebenfalls wichtig, ist aber auch oft neu und gehört nicht zur Routine. Ggf. kommt es einen auch etwas albern vor. Übungen dazu findet man für jeden Geschmack und angepasst auf die eigene physischen Möglichkeiten im Internet (Suche: „Arbeitsplatzergonomie bewegte Pausen")

Spielregeln für Videokonferenzen und co

Videokonferenz-Services, wie Webex, GoToMeeting, Microsoft Teams aber auch eine Vielzahl kleinerer Anbieter wie https://8x8.vc sind am Markt. Generell sollten diese Services folgende Mindeststandards haben: Ausreichend viele Benutzer unterstützen, Chat, Screensharing und Wechsel auf die Screens der Teilnehmer, Mute, optionaler Audio Dial-in (mit Telefon) und Meeting Aufzeichnung.

Bei Videokonferenzen helfen folgende Punkte, damit das Meeting gut und brauchbar ist:

Vorbereitung

- Pünktlich Konferenz eröffnen, pünktlich einwählen, pünktlich beenden
- Ruhige Umgebung suchen
- Vorbereitung der Agenda - und vorher Lesen (durch die Teilnehmer)
- Alle: Bitte sorgen Sie auch für eine ordentliche Infrastruktur - ein Laptopmikrophone ersetzt kein Jabra! Ggf. auf Headset setzen
- Festlegen, wer der Moderator bzw. Facilitator ist; damit gibt es jemand, der Ablauf und Ziele des Meetings verantwortet
- Moderator: Test der Infrastruktur, spätestens 10-15 min vor dem Start der Konferenz und damit sicherstellen, dass das Videokonferenz-Service up-and-running ist; wenn der vorzubereitende Termin wirklich wichtig ist, empfiehlt sich im Kontext auch eine Alternative in der Hand zu haben. Nicht selten hatten in den letzten Tagen einzelne Anbieter, insbesondere auch Big Player, Performance Probleme, die sich massiv auf das Meeting ausgewirkt haben
- Teilnehmer: Es ist einfach waste, wenn alle warten müssen, bis die notwendig PlugIns und co am PC geladen sind. Bitte pünktlichen Start absichern

Darüber hinaus ist eine ausreichende Internetverbindung wichtig. Falls Sie zu wenig Bandbreite haben: Nutzen Sie den Audio Dial-In mit dem Telefon und vermeiden Sie jedenfalls Video-Sharing. Trotzdem, wenn irgendwie möglich: Nutzen Sie Screensharing - dies ist für die Synchronisierung der Teilnehmer und für einen effizienten Meetingverlauf wichtig.

Wenn ausreichend Bandbreite zur Verfügung steht: Dann bitte unbedingt alle Gesichter der Videokonferenz zeigen. Damit sind alle „am Ball" und echt eingebunden. Fordern Sie die Teilnehmer auf nahe bei der Kamera zu sein, damit man die Gesichter auch wirklich gut sieht - damit simuliert man ein physisches Meeting am Besten.

Generell sollten Sie auch diese Infrastruktur testen / üben: Nutzen Sie einfach eines Ihrer kommenden Meetings und testen Sie Infrastruktur, Ablauf, Technik und Regeln.

In der Videokonferenz

Idealerweise protokollieren Sie online: Vorausgesetzt, dies kann ausreichend rasch gemacht werden. Dokumentieren Sie jedenfalls auch pro Abschnitt vereinbarte ToDo's, wiederholen Sie diese auch am Ende des Meetings.

Wenn Videokonferenz, dann sollten alle Teilnehmer via Videokonferenz arbeiten - das bedeutet, dass sich jeder einwählt; damit stellen Sie sicher, dass alle Teilnehmer gleich behandelt werden. Das wird uU nicht immer funktionieren / praktikabel sein. Für diese Fälle bitte sicherstellen, dass alle im Gespräch eingebunden sind - und reden Sie bitte in Richtung Mikrophone.

Weitere Spielregeln:

- Stellen Sie das Mikrophone auf Mute, wenn Sie gerade nicht dran seid
- Fragen „Passt das für alle" sollten gedreht werden „Ist jemand dagegen?" - damit werden Pausen in der Kommunikation reduziert
- Videokonferenzen sind keine Multitasking Veranstaltungen: Beschäftigen Sie sich nicht mit anderen Dingen (Emails lesen und dgl.)
- Als Moderator
  - Klar durch die Agenda führen, Disziplin einfordern und immer wieder mal „alle" einbinden (damit niemand abtriftet), vielleicht auch Abstimmungen reihum machen und die einzelnen Teilnehmer konkret um Zustimmung & Feedback zu einzelne Punkte bitten. Dies kann auch durch entsprechende Polls gemacht werden um insbesondere bei größeren Gruppen Zeit zu sparen
  - Und zum Meetingstart kurz eine soziale Aufwärmphase (1-5 min) einplanen. Einfache Fragen zum Befinden, aber auch zu aktuellen Einflüssen auf das eigene Unternehmen/Arbeitsweise sind Standards, die man aus den physischen Treffen kennt und die auch hier wichtig sind
  - Next steps, ggf. nächste Termine fixieren
- Teilen Sie die jeweils relevanten Bildschirme - ein Bild sagt oft mehr wie tausend Worte; berücksichtigen Sie jedoch, dass selbst auf großen Displays mehr wie 3 Bildschirme im Detail selten ausreichend sichtbar sind
- Reduzieren Sie Satzlängen auf das nötige Maß - weniger ist mehr; gerade, wenn non-verbale Kommunikation reduziert ist, ist es schwer allen Inhalten zu folgen

Und hier noch zum Abschluss:

Weniger ist mehr - Online Konferenzen über 90 min sind schon problematisch und reduzieren mit Sicherheit die Produktivität.

Gerade, wenn man sich nicht sieht, sind ToDo-Tools, wie Jira und co ein Muss! Stellen Sie bitte auch sicher, dass ToDo's gelogged, aktualisiert und abgearbeitet werden.■

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.

Agile Circle

Alexander weixi Weichselberger hat den Agile Circle ins Leben gerufen. Der Agile Circle ist der Treffpunkt für Umsetzerinnen und Umsetzer digitaler Innovationen. Hier können sich agile Führungskräfte, Softwareentwicklerinnen und Softwareentwickler, Testerinnen und Tester, Product Owner und Scrum Master treffen, austauschen, inspirieren und neue Ideen entwickeln.

Der Agile Circle bringt die unterschiedlichsten Blickwinkel auf die agile Software Entwicklung zusammen: das Management, die Entwicklung, sowie das Testing & die Delivery.

Mit Stand März 2020 bietet der Agile Circle eine Reihe von Veranstaltungen und Community Treffs, diese sind:

Agile Circle ONLINE - 28.05.2020
Agile Circle UNCONFERENCE - Herbst 2020
Agile Circle CONFERENCE - 27.05.2021

Weitere Informationen finden Sie auf unserer Website: www.agilecircle.org

Schauen Sie sich unser YouTube Video zum Thema Agile Circle an:

5 Tipps für ein sicheres Passwort

von *Klemens Loschy*

Am 01.02.2020 war der „Ändere dein Passwort" - Welttag

... allein, dass dieser Tag überhaupt existiert beweist, dass man gar nicht oft genug über das Thema „sichere Passwörter" reden kann. Eine zyklische Änderung von Passwörtern ist nicht mehr zeitgemäß und führt, wenn es z.B. vom Unternehmen forciert wird, nicht zu mehr Sicherheit: „MeinPasswort1" ist genauso sicher oder unsicher wie „MeinPasswort2", „MeinPasswort3" usw. Vielmehr gelten heute andere Regeln, also noch vor ein paar Jahren:

SEQIS GmbH

1. Verwende für jeden Dienst ein anderes, generiertes Passwort

Mittlerweile muss man sich gefühlt überall mit seiner EMail Adresse registrieren, und vergibt genauso oft ein Passwort. Dieses Passwort muss in Kombination mit der EMail Adresse eindeutig sein (am Besten mit mehr als 16 Zeichen, bestehend aus Buchstaben, Ziffern und Sonderzeichen)! Der Hintergrund ist einfach: werden Ihre Zugangsdaten bei einem Dienst geknackt oder gestohlen, so ist wirklich nur dieser eine Dienst betroffen. Alle anderen Dienste sind weiterhin sicher! Es hilft also das sicherste Passwort nichts, wenn es (aus welchen Gründen auch immer) bekannt wurde und daraufhin all Ihre Dienste zugänglich sind.

2. Verwende einen Passwortmanager

Eindeutige (generierte) Passwörter haben natürlich einen großen Nachteil: das kann sich einfach keiner merken. Das ist aber auch gar nicht notwendig: Passwortmanager machen nämlich genau das, sie merken sich Passwörter (und generieren sie bei Bedarf auch). Genauer gesagt werden darin Zugangsdaten zu all Ihren Diensten verwaltet. Mittlerweile gibt es sehr viele Passwortmanager mit unterschiedlichsten Features, kostenpflichtig und kostenfrei. Mit LastPass, Dashlane, Bitwarden und 1Password seien nur ein paar davon genannt. Der Zugang zum Passwortmanager selbst ist natürlich wieder mit einem Passwort, dem s.g. Masterpasswort, gesichert. Das muss besonders stark und gleichzeitig aber auch einprägsam sein, denn das müssen Sie sich merken!

3. Verwende Passphrases

Wenn es wirklich notwendig ist sich sichere Passwörter zu merken, geht die Empfehlung klar weg vom Passwort und hin zur Passphrase: das ist im weiteren Sinn einfach ein normaler Satz. Er darf aber nicht aus einem Buch, Lied, Gedicht, Film usw. stammen. Am besten ist, wenn er keinen Sinn ergibt: „Ein kalter Tag ist kürzer als 2 dunkle Nächte, das weiß doch jeder!" - sinnlos, aber sicher und einprägsam und eignet sich daher z.B. als Masterpasswort für Ihren Passwortmanager.

4. Verwende Biometrische Hilfsmittel

Die meisten Smartphones bieten die Möglichkeit statt der Eingabe von Passwörtern oder PINs die integrierte Fingerprint oder Gesichtserkennung zu verwenden. Ebenso unterstützen das immer mehr Notebooks. Verwenden Sie diese Hilfsmittel z.B. als Masterpasswort für Ihren Passwortmanager. Das eigene Gesicht muss man sich nicht merken, verlieren kann man es auch nicht und nachmachen ist wirklich schwierig. Die Sicherheit dieser Features wird immer wieder von IT Security Firmen getestet mit dem Ergebnis, dass eine Umgehung nur mit enormem Aufwand möglich ist. Für den aller größten Teil der Nutzer ist das eine massive Verbesserung.

5. Verwende 2 Faktor Authentifizierung

Viele Dienste bieten die Möglichkeit einer 2 Faktor Authentifizierung (2FA oder TFA) an. Dabei wird zusätzlich zum Passwort noch ein zweiter Faktor zur Authentifizierung verwendet. Die meisten kennen das vermutlich vom Online Banking, wo eine Überweisung zusätzlich durch einen per SMS versendeten Code autorisiert werden muss. Das SMS System wurde von s.g. OTP (One Time Passwords) und den passenden Apps (z.B. Google Authenticator oder OTP Auth) abgelöst. Das eigene Handy wird mit Hilfe der OPT App beim jeweiligen Dienst registriert und ab da an muss man nach der Authentifizierung mit EMail und Passwort ein zusätzliches generiertes Einmal-Passwort (meist sind das 6 Ziffern) angeben, das die OTP App am Handy anzeigt. Nach der Eingabe ist dieses Einmal-Passwort ungültig, daher auch der Name. Selbst wenn also die Zugangsdaten bekannt sind ist ein Login nur dann möglich, wenn der Angreifer zusätzlich auch mein Handy hat.

Fazit

Es macht also keinen Sinn, eindeutige generierte Passwörter zyklisch zu ändern. Genausowenig macht es Sinn zyklisch eine starke Passphrase zu ändern. Eine Änderung macht natürlich dann Sinn, wenn ein von mir verwendeter Dienst kompromittiert wurde: viele große und bekannte Unternehmen „verlieren" immer wieder Zugangsdaten ihrer Benutzer (das ist weit wahrscheinlicher als dass mein Passwort durch Angreifer geknackt wird) und davor schützt uns nicht das sicherste Passwort, auch dann nicht, wenn wir es zyklisch ändern. Das Gute ist: mit dem eindeutigen Passwort kann man sich sicher sein, dass mit den Zugangsdaten keine anderen Dienste verwendet werden können.

Ich selbst verwende erst seit ca. 3 Jahren einen Passwortmanager. Der Anfang ist aufwändig, denn jeder von mir verwendete Dienst musste auf ein neues generiertes Passwort geändert werden. Jetzt möchte ich den Komfort aber nicht mehr missen und ich weiß, dass dadurch meine Daten und Zugänge viel besser geschützt sind als zuvor!

Macht der „Ändere Dein Passwort Tag" Sinn? Er regt zu mindest zu einer Diskussion rund um das Thema „Sichere Passwörter" an, man sollte den Tag aber jedenfalls nicht zu wörtlich nehmen.

Mehr spannende Blogartikel finden Sie hier:

https://www.seqis.com/de/blog-index

Schauen Sie sich gerne unser YouTube Video zum Thema „Ändere dein Passwort - Welttag" an.

Klemens Loschy ist Principal Consultant, Teamlead bei SEQIS.

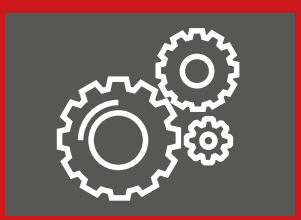
Er kann auf jahrelange Erfahrung in den Bereichen Testautomation, Last-Tests und Performance Engineering, funktionale Tests, Testen in agilen Teams, Anwendungsentwicklung von Testsoftware sowie Beratung und Unterstützung in zahlreichen Projekten unterschiedlichster Branchen zurückblicken.

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.

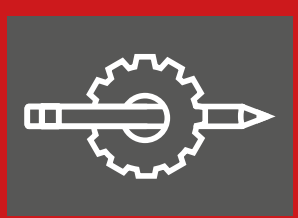
IT ANALYSE
Notwendige Änderungen analysieren und IT-gerecht aufbereiten

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