FALSCHE ANNAHMEN UND IHRE FOLGEN

Mythen der Softwareentwicklung

Führungskräfte treffen oft Entscheidungen, die auf unrealistischen Annahmen beruhen.

Schon seit einiger Zeit begegnet mir ein Thema als Berater und Trainer immer wieder: die falschen Annahmen von Situationen in der Softwareentwicklung. Auf Basis dieser Annahmen treffen Führungskräfte meist Entscheidungen, die fatale Folgen für Projekt und Unternehmen haben. Das Merkwürdige ist: Diese Mythen begegnen mir jeden Tag aufs Neue. Dieser Artikel soll mit einem Teil davon aufräumen und sieben dieser Mythen entmystifizieren. Dabei ist mir bewusst, dass diese Mythen bei Weitem nicht die vollständige Legendenbildung abdecken, aber sie sind ein Anfang.

Mythos 1: Unit-Tests sind das Allheilmittel

Das Testen von Software ist ohne jede Frage sehr wichtig für ein qualitativ hochwertiges Endprodukt. Oft gilt dies als die letzte Rettung in einem Projekt, das in Schieflage geraten ist. Ich habe schon viele Projekte begleitet, bei denen der Abteilungsleiter oder der Geschäftsführer beschlossen hat, dass die fehlerhafte Software mit Unit-Tests abgesichert werden soll, um der Lage Herr zu werden. Das Ziel ist gesetzt, die Umsetzung dieser gut gemeinten Idee ist nun Aufgabe der Entwickler – so weit, so gut. Aber löst das wirklich die Probleme einer fehlerhaften Anwendung? Als passionierter Fan des Testens von Software glaube ich das nicht.

Zunächst stellt sich die Frage, warum die Software voller Fehler ist. Bei der Analyse des Problems zeigen sich oft folgende Ursachen:

- schlechte Anforderungen,
- hoher Zeitdruck,
- komplexer Code,
- unzureichende Ausbildung.

Was bringt es nun also, die Software zu testen? Wenn eine der oben genannten Ursachen auf das Problem zutrifft, mag das Erstellen von Unit-Tests womöglich eine geringe Abhilfe schaffen, aber es ist garantiert keine Lösung für das ursprüngliche Problem. Anstatt wertvolle Ressourcen für das Testen aufzuwenden, sollte stattdessen lieber das Ursprungsproblem gelöst werden. Und das geht in der Regel wesentlich schneller und damit auch billiger.

Darüber hinaus ist auch zu erwähnen, dass das Schreiben von Unit-Tests einige Anforderungen an das Projekt stellt, zum Beispiel:

- ein entkoppeltes Design,
- gute Modularisierung,
- genug Ressourcen,
- Know-how über Unit-Tests.

Um es auf den Punkt zu bringen: Das Schreiben von Unit-Tests ist teuer, verlangsamt die Entwicklung und setzt eine ordentliche Struktur voraus. Ist Letztere nicht vorhanden, muss die Struktur sehr aufwendig umgebaut werden. Können Sie sich das wirklich leisten, oder sollten Sie nicht lieber die Ursache bekämpfen?

Mythos 2: Architektur kommt von alleine

Viele meiner Kundenprojekte haben die Architektur von Anwendungen zum Thema. Meist wird erst dann ein Berater hinzugezogen, wenn das Kind schon ziemlich tief in den Brunnen gefallen ist. Nach einer gründlichen Analyse ist das Ergebnis für die Führungskräfte oft ein ordentlicher Schock, und die monetäre Bezifferung des Schadens stellt Projekte und Unternehmen nicht selten vor massive Probleme.

Interessanterweise höre ich dabei immer wieder Sätze wie „Ich dachte, die Entwickler kennen sich mit Architektur aus“ oder „Die Architektur sollte doch von alleine kommen“. Wie entstehen diese Annahmen? Wie jedes andere Thema in einer Ingenieursdisziplin – wie der Softwareentwicklung – muss auch das Thema Architektur gelernt und somit geschult werden. Es gibt nur sehr wenige Fälle, bei denen die Ausbildung für studierte oder gelernte Entwickler auch die Themen Softwarequalität oder gar Architektur enthält. Folglich muss dieses Wissen im Projekt aufgebaut werden, entweder durch Trainings oder durch autodidaktisches Lernen. Kein Fehler in einem Softwareprojekt ist in seiner Behebung so teuer wie eine fehlerhafte Architektur, da diese eine sehr kostspielige Refaktorisierung zur Folge hat. Dabei ist klar, dass nicht in jedem Projekt ein eigener (Software-)Architekt zur Verfügung steht. Aber zumindest sollte bei den führenden Entwicklern im Team genügend Kenntnis darüber vorhanden sein, um das restliche Team anzuleiten.

Mythos 3: Der Entwickler kann Anforderungen aufnehmen

Wie bereits ausführlich diskutiert, werden einem Entwickler viele Talente nachgesagt [1]. Während seine Begabung zweifelsohne sehr ausgeprägt ist, wenn es um technisches Verständnis und die Umsetzung davon geht, trifft dies in den meisten Fällen nicht auf das Sammeln, Abstimmen und Ver-

feinern von Anforderungen zu. Es dürfte nur wenige Entwickler geben, die dies wirklich sehr gut können und vor allem auch gerne machen. Das soll nicht die Rolle der Person, welche die Anforderungen aufnimmt, in irgendeiner Art und Weise schmälern, sondern nur die Realität aufzeigen.

Für technisch interessierte Menschen ist das Sammeln von Geschäftsanforderungen eines Projekts nicht sehr spannend, und so ist die Qualität der Anforderungen, die sich daraus ergibt, oft schlecht. Darüber hinaus ist es ja nicht so, als ob jeder im Projekt so ohne Weiteres Anforderungen aufnehmen kann und diese auf einem qualitativ guten Niveau optimieren, verfeinern und in einem Ticketsystem ablegen kann. Dazu wird neben einem guten Domänenverständnis und einer guten Kommunikationsfähigkeit genauso ein Talent vorausgesetzt wie für das Entwickeln von Software.

Ebenso wie die Architektur sind Fehler in der Anforderung teuer. Folglich sollten diese also mit dem höchstmöglichen Grad an Qualität aufgenommen werden. Liebe Mitentwickler, seid mir nicht böse, aber diesen Grad erreichen wir nur in den seltensten Fällen. Anforderungen sollten von dafür ausgebildeten Fachleuten aufgenommen und dem Entwickler in aufbereiteter Form übergeben werden.

Mythos 4: Scrum löst alle Probleme

Software wird oft ohne fest definierten Prozess entwickelt [2]. Die dadurch auftretenden Probleme sollen dann meist durch ein geeignetes Prozessmodell oder -Framework gelöst werden. Nicht selten besteht die vermeintliche Lösung dabei im Einsatz von Scrum. Das Merkwürdige dabei ist, dass dieser Wunsch oft nicht von den Entwicklern, sondern von den Führungskräften geäußert wird. Dabei ist das sicher eine gute Lösung, denn Scrum eignet sich bei entsprechenden Voraussetzungen sehr gut für die Softwareentwicklung. Aber sind diese Voraussetzungen nicht gegeben, ist das Einführen von Scrum nutzlos und unter Umständen sogar kontraproduktiv. Wenn Anforderungen nicht sauber formuliert sind, kann auch Scrum keine besseren Ergebnisse erzielen. Wenn Teams im Sprint mit weiteren Arbeiten belastet werden, erreichen sie das Sprintziel nicht; wenn keine ernsthafte Retrospektive abgehalten wird, wird aus Scrum nie ein optimierter Prozess, sondern es bleibt ein loses, schlechtes agiles Vorgehen.

Auch hier will ich nicht falsch verstanden werden: Scrum ist eine exzellente Umgebung, um Softwareprojekte durchzuführen. Aber zur richtigen Umsetzung müssen auch die Rahmenbedingungen passen. Ist das nicht der Fall, scheitert Scrum kolossal. Werden diese Rahmenbedingungen nicht eingehalten, so wird nicht nach Scrum gearbeitet, sondern höchstens agil. Der Satz „Wir machen etwas Scrum-ähnliches“ ist fast schon ein Running Gag.

Mythos 5: Desktop auf Web ist einfach

Die Zukunft des Desktops ist nicht besonders rosig. Das Dasein sehr vieler Anwendungen liegt im Web, und der Desktop wird maximal noch eine geringfügige Nebenrolle für Spezialanwendungen spielen. Aber was passiert mit den bestehenden Anwendungen und den vielen WPF- oder Windows-Forms-Entwicklern da draußen? „Wir schulen, und danach stellen wir auf Web um“ ist eine der häufigsten Antworten von Vorgesetzten. Realistisch betrachtet ist das die einzig mögliche Antwort, allerdings muss man sich auch der Konsequenzen bewusst sein. In der Webentwicklung existieren zum Teil vollkommen andere Konzepte als in der Entwicklung von Desktop-Anwendungen.

Als wären diese Konzepte nicht alleine schon Herausforderung genug, sind auch die Tools, Technologien und Frameworks komplett neu, und das dazugehörige Ökosystem ist nochmals eine vollkommen andere Welt. Das heißt: Während ein Desktop-Entwickler im Backend noch auf einen Teil seines Wissens aufbauen kann, ist die Welt des Frontends komplett neu. Sie wollen eine moderne Webanwendung bauen? Dann müssen Ihre Entwickler HTML, CSS, JavaScript und TypeScript lernen, um überhaupt etwas auf den Bildschirm zu zaubern. Damit das auch noch gut aussieht, muss auch ein CSS-Framework wie Bootstrap verstanden und angewendet werden. Aber um wirklich modern zu sein, sollte lieber Sass oder LESS mit Angular 2 und den vielen damit verbunden JavaScript-Frameworks wie Webpack, RxJS oder sonstigen verwendet werden. Und um es erträglich zu gestalten, sollte auch Gulp oder Grunt als „Build-System“ im Client genutzt werden. Das wäre dann auf der Höhe der Zeit.

In all dem müssen die Entwickler geschult werden, was mit Kosten verbunden ist und viel Zeit erfordert. Kein Entwickler kann 30 Tage am Stück verschiedene Sprachen und Frameworks lernen und diese sofort anwenden. Webentwicklung verlangt viel Wissen und besonders Erfahrung – beides stellt sich erst im Lauf der Zeit ein. Auch das Alter des Entwicklers kann ein Faktor sein. Ein großer Erfahrungsschatz aus der Desktop-Entwicklung bringt hier meist nicht viel.

Ist es also unmöglich? Nein, aber seien Sie realistisch: Eine Umstellung von Desktop- auf Webentwicklung ist mit entsprechenden Kosten verbunden und geht nicht von heute auf morgen. Viele Projekte haben jedoch keine Alternative.

Mythos 6: Jeder soll alles können

Risiken in der Softwareentwicklung gibt es viele, und meist ebenso viele Maßnahmen, um sie zu minimieren. Wenn es um Themen wie „Wissensinseln“, „Truck-Faktor“ oder Ausfallsicherheit von Entwicklern geht, wird oft die Maxime „jeder soll alles können“ ausgerufen. Soll heißen: Jeder Entwickler soll in der Lage sein, jede anfallende Aufgabe in einem Projekt lösen zu können. In Einzelfällen ist dies durchaus eine praktikable Lösung, aber wie immer sollte der Kontext betrachtet werden. Sind die Projekte immer gleich? Wie groß ist die Domäne? Wie vielfältig die Technologien?

Einer meiner Kunden verfolgt genau dieses Prinzip, und bei 30 Entwicklern sollen alle Entwickler alles können. Dazu zählen Desktop, Web, Backend, Services, SQL, Deployment, Testen und vieles mehr. Klappt es? Natürlich nicht. Die Entwickler sind überfordert, die Fehler häufen sich, die Zufriedenheit der Kunden nimmt ab und die Qualität der technischen Umsetzung sinkt. Aufgaben in schwierigen Bereichen bleiben lange liegen und werden dann im Endeffekt durch teure externe Firmen umgesetzt, weil die nötige Expertise im eigenen Team einfach verloren gegangen ist. Die Mitar- ▶

beiter sind hochgradig unzufrieden, und nahezu jeden Monat verlässt ein Mitarbeiter das Team – das damit eine Menge Know-how über bestehende Systeme verliert. Natürlich müssen die eingangs erwähnten Risiken minimiert werden, und mit dieser Methode mag dies auch in Ausnahmefällen möglich sein, aber in allen anderen Fällen ist es die schlechteste mögliche Lösung und schafft mehr Probleme, als sie löst.

Mythos 7: Jeder kann Software entwickeln

Zum Schluss sei noch ein Mythos genannt, der mir persönlich sehr am Herzen liegt, weil keiner der zuvor genannten dermaßen unwahr ist. Als Trainer schule ich im Jahr Tausende Entwickler in den unterschiedlichsten Bereichen und kenne gute und schlechte, fleißige und faule, unbegabte und talentierte Entwickler aller Altersstufen. Dabei kann man durchaus provokant die Frage stellen: Wann ist ein Entwickler gut?

Gute Entwickler zeichnen sich meiner Meinung nach meist durch eine oder mehrere der folgenden Eigenschaften aus:

- Talent,
- Gründlichkeit,
- Wissbegierigkeit,
- Neugier,
- Lernbereitschaft,
- Kritikfähigkeit,
- Leidensfähigkeit,
- Teamfähigkeit.

Während sich einige der Punkte erlernen lassen, sind die meisten jedoch quasi „von Gott gegeben". Ohne Begeisterung und Passion für Technik und speziell die Softwareentwicklung kann niemand die genannten Punkte in der Softwareentwicklung über längere Zeit aufrechterhalten. Er wäre auch kein guter Softwareentwickler, und es ist stark zu bezweifeln, dass er in seinem Beruf auf Dauer wirklich Glück und Erfüllung finden würde. Besonders stört an dem letzten Mythos, dass diese Behauptung schlichtweg falsch ist und von Verblendung zeugt.

Ich habe vor einiger Zeit bei einem langjährigen Kunden die Ausbildung im Bereich Softwareentwicklung übernommen, damit alle Auszubildenden nach der erfolgreichen Abschlussprüfung möglichst schnell in Projekten eingesetzt werden konnten. Alle Azubis bis auf einen gaben sich Mühe, arbeiteten nach und beherrschten ein Thema meist schon am Schulungstag gut genug, um es in Praxisprojekten einzusetzen. Diese Azubis schafften die Ausbildung, der eine jedoch fiel mehrfach durch die Abschlussprüfung und brach die Ausbildung daraufhin ab. In einem Gespräch sagte er mir, dass seine Entscheidung zu diesem Beruf darauf beruhte, dass bei der Ausbildungsberatung seine Neigung zu Computerspielen erkannt wurde und der Berufsberater auf der Messe ihm gesagt hätte, Programmieren wäre leicht und könnte von jedem erlernt werden.

Ich vermute, jeder Leser dieser Kolumne kann in irgendeiner Weise programmieren, sei es C#, F#, C++, C, JavaScript, PHP oder eine sonstige Sprache. Ich denke also, jeder von Ihnen kann nachvollziehen, wenn ich nur entsetzt den Kopf schüttelte.

Softwareentwicklung ist spannend, herausfordernd, frustrierend, anstrengend und macht unwahrscheinlich viel Spaß, auch nach so vielen Jahren noch genauso wie am ersten Tag – aber sie ist ganz bestimmt nicht einfach und kann definitiv nicht von jedermann erlernt werden.

Ein Softwareentwickler lernt jeden Tag, und das sein gesamtes Arbeitsleben lang. Nahezu jedes Projekt ist anders, ständig muss er sich neue Technologien aneignen und sie anwenden – und ich kenne viele, die das jeden Tag aufs Neue mit absoluter Begeisterung und Hingabe machen. Viele gehen sogar noch nach Feierabend und im Urlaub dem jeweiligen Thema mit Begeisterung nach, vollkommen freiwillig. Wie viele Steuerberater kennen Sie, die nach Feierabend noch voller Freude und Begeisterung eine Steuerklärung aus Neugierde ausfüllen?

Jeder kann Programmieren lernen, dem stimme ich zu. Aber dazu sind Fleiß, Motivation, Ausdauer und eine hohe Frustrationsgrenze nötig – und das alles ist ganz bestimmt nicht einfach.

Fazit

Dem Leser wird nicht entgangen sein, dass ich innerhalb des Artikels mehrfach Bezug auf die Führungskräfte genommen habe. Ich mache sie keineswegs für die Fehlentscheidungen verantwortlich, die sie auf Basis dieser Mythen getroffen haben. Jeder verantwortungsvolle Mensch trifft seine Entscheidungen auf einer soliden Grundlage, das gilt insbesondere für diese Führungskräfte. Fehlt einem Entscheider diese Grundlage, sollte er entweder auf die eigenen Mitarbeiter hören [3] oder einen externen Berater hinzuziehen.

Ziel soll lediglich sein, alle interessierten Leser für die sieben hier vorgestellten Mythen zu sensibilisieren und hoffentlich aufzuzeigen, dass sie eben keine solide Grundlage für eine Entscheidung sind. Betrachten Sie das hier Geschriebene einfach als eine Art Antimuster für Entscheidungen in der Softwareentwicklung – lernen Sie aus den Fehlern anderer. ■

[1] David Tielke, Richtig planen, Was machen Entwickler eigentlich alles? dotnetpro 8/2017, Seite 72 ff., www.dotnetpro.de/A1708DDD

[2] David Tielke, Wie gut ist Ihr Prozess? Wie ein Prozess aussehen sollte, dotnetpro 7/2017, Seite 58 ff., www.dotnetpro.de/A1707DDD

[3] David Tielke, Auf Mitarbeiter hören, Kontinuierlich verbessern, Teil 1, dotnetpro 9/2017, Seite 54 ff., www.dotnetpro.de/A1709DDD

David Tielke
ist freiberuflicher Berater und von Microsoft zertifizierter Trainer für die Anwendungsentwicklung auf der .NET-Plattform. Darüber hinaus hat er sich auf die Bereich Softwarearchitektur, Softwarequalität und ALM spezialisiert.
mail@david-tielke.de

dnpCode A1711DDD