Als Entwickler glücklich sein – Tipps und Tricks

Christian Seifert, BetterDoc

Der Softwareentwickler, der sein Hobby zum Beruf gemacht hat – so würden sich sicherlich viele von uns charakterisieren. Doch angekommen im Beruf sieht die Welt nicht selten plötzlich gar nicht mehr so rosig aus. Aus den verschiedensten Gründen sind wir als Entwickler gefrustet und denken uns: „So hatte ich mir das aber nicht vorgestellt!". Häufig liegt das an unseren Erwartungshaltungen und Vorstellungen, die mit der Realität nicht oder nur wenig übereinstimmen. In diesem Artikel wollen wir uns einige „Real Life Happiness Patterns" ansehen, die dabei helfen können, den ganz normalen Alltagswahnsinn ein wenig leichter zu überstehen und den Spaß an der eigenen Arbeit wiederzufinden.

Der typische Softwareentwickler sitzt spät abends allein im spärlich beleuchteten Keller, neben ihm die leere Pizzabox und zwei Flaschen Cola. Die Realität von professioneller Softwareentwicklung hat mit dieser idealisierten Vorstellung nur sehr, sehr wenig zu tun. Und dennoch ist es dieses Bild, das wir häufig im Kopf haben und das für viele eine Idealvorstellung ist: Ungestört vom Rest der Welt, nur fokussiert auf die Technik, durch das Schreiben von Code kniffelige Probleme lösen. Stattdessen haben wir ganz andere Herausforderungen: quengelige Kunden, fordernde Projektmanager, enge Zeitpläne und Kollegen, die vollkommen andere Vorstellungen davon haben, was am Ende herauskommen soll, als man selbst. Das Ergebnis ist nicht selten Frust, Resignation und in immer mehr Fällen auch Burnout.

Mit einem frischen Blick auf die eigene Umwelt und das eigene Selbstverständnis finden sich aber durchaus Auswege, den Spaß und die Freude an der Arbeit wiederzufinden und Softwareentwicklung nicht nur als notwendiges Übel zu sehen, das am Ende des Monats dafür sorgt, dass Geld auf dem eigenen Konto ist, sondern es wieder als spannende Herausforderung zu sehen, für die man gerne morgens aus dem Bett steigt.

Erwartungen und die Realität

Wie in vielen anderen Bereichen gibt es auch von Softwareentwicklung ein idealisiertes Bild, das mit der Realität nur selten übereinstimmt. Eine grundlegende Frage, die wir uns alle stellen sollten, ist daher: Was will ich eigentlich wirklich in meinem Job? Womit möchte ich mich beschäftigen und (vielleicht ebenso wichtig) womit möchte ich mich nicht beschäftigen? Nur wer für sich selbst definiert hat, worauf er seinen Fokus legen möchte und wo seine Grenzen liegen, kann aktiv nach eben den Dingen suchen, die für ihn selbst interessant sind.

Gerade der bereits beschriebene Weg vom Hobby zum Beruf ist häufig eher hinderlich als förderlich: Software allein in der Freizeit zu schreiben erlaubt (und erfordert) eine andere Denk- und Herangehensweise, als Software im Team mit Kollegen für Geld zu entwickeln und diese zu warten. Als Tipp gilt daher: „Wer sein Hobby zum Beruf gemacht hat, tut gut daran, sich als Erstes ein neues Hobby zu suchen". Ein Hobby sollte nicht zuletzt auch dazu dienen, abzuschalten und den Kopf freibekommen zu können. Wer sich aber sowohl beruflich als auch in seiner Freizeit mit den immer gleichen Dingen beschäftigt, für den wird es zunehmend schwieriger, den „Akku aufzuladen".

Stakeholder kennen und verstehen

Professionelle Softwareentwicklung ist bereits seit Langem keine One-Man-Show mehr. Wir finden uns eigentlich immer in einem Team mit den verschiedensten Rollen und Persönlichkeiten wieder. Viele der größten Missverständnisse und Enttäuschungen entstehen für uns als Softwareentwickler dadurch, dass wir davon ausgehen, alle anderen haben die gleiche Denkweise, verfolgen die gleichen Ziele und nutzen die gleichen Methoden wie wir selbst. Interesse, über den eigenen Tellerrand blicken zu können und zu wollen, ist hierbei einer der Schlüssel zu einer besseren, reibungsloseren und damit stressfreieren Zusammenarbeit. Nur wer wirklich versteht, wie sein Gegenüber tickt, was seine Ziele und Motivationen sind, der wird in der Lage sein, auf Augenhöhe zu kommunizieren und seine eigenen Wünsche und Probleme zu äußern.

Viele Projekte, nicht nur, aber auch und gerade im Software-Umfeld, scheitern nicht an den technischen Rahmenbedingungen. Sie scheitern nicht daran, dass man das falsche Framework oder die falsche Datenbank ausgewählt hat, sondern daran, dass das eigentliche Problem nicht oder nur schlecht verstanden wurde. Die Fähigkeit und die Bereitschaft, die Sprache unseres Gegenübers zu verstehen und aktiv sprechen zu können, ist dabei einer der größten Hebel, die wir haben, um auf unsere Probleme aufmerksam zu machen und unsere Sichtweise zu artikulieren. Wird allein mit technischem Jargon versucht, einen Fachverantwortlichen davon zu überzeugen, Dinge zu ändern oder Probleme anzugehen, der wird sich früher oder später frustriert zurückziehen, weil „der mir sowieso nicht zuhört und nicht versteht, was ich ihm zu sagen versuche".

Hier hilft es, sich selbst in einer ähnlichen Situation zu beobachten: Stellen wir uns vor, nach einem Sportunfall bekommen wir von einem Arzt zu hören, dass wir uns eine Radiuskopffraktur, eine Scaphoidfraktur und eine laterale Tibiakopffraktur zugezogen haben. Vollkommen zu Recht würde jeder von uns erwarten, diese Diagnose für einen Nicht-Mediziner verständlich übersetzt zu bekommen: Knochenbrüche im Ellenbogen, der Handwurzel und seitlich am Schienenbein. Ebenso sollten wir als Softwareentwickler in der Lage sein, unsere Themen für Nicht-Fachleute übersetzen zu können.

Die Konsequenzen sind hierbei enorm und direkt zu spüren: Nur wenn andere Projektbeteiligte unsere Probleme verstehen und nachvollziehen können, sind sie auch in der Lage, uns unterstützen zu können, was uns wiederum in die Lage versetzt, unsere eigene Arbeit besser, leichter und motivierter zu erledigen.

Wertschätzung der eigenen Arbeit

Die Softwareindustrie entwickelt sich mit einer atemberaubenden Geschwindigkeit. Technologien, die buchstäblich gestern noch als hochinnovativ und fortschrittlich galten, sind heute schon überholt. Doch nicht jedes Projekt und nicht jede Softwarelösung kann (und will) mit diesem enormen Tempo mithalten. Viele Banken und Versicherungen entwickeln heute noch aktiv mit Technologien, die ihre Ursprünge in den 1950er und 1960er Jahren haben. Wer einmal in die großen Stellenportale schaut, wird verblüfft sein, wie viele Stellenangebote es auch 2019 noch für COBOL-Entwickler gibt.

In Entwicklerkreisen werden solche Legacy-Anwendungen gerne belächelt und Entwickler, die nicht ganz vorne auf der Innovationswelle reiten, als zweitklassig eingestuft. Wer nicht mit den neusten Technologien umgehen kann, wird ungern als vollwertiges Mitglied des Clubs angesehen.

Doch sollten wir dabei nicht vergessen, dass viele dieser Altanwendungen sicherstellen, dass essenzielle Bestandteile unseres täglichen Lebens einwandfrei funktionieren. Natürlich mag ein System zur Verwaltung von Rentenversicherungen von außen eher langweilig und verstaubt wirken, doch wird jeder von uns hoffen und davon ausgehen, dass seine Versicherung Sorge dafür trägt, die ihr anvertrauten Gelder korrekt zu verwalten. Ein Teil des Entwicklerteams zu sein, das eben diese Software pflegt und weiterentwickelt, sollte daher nicht per se dazu führen, als Entwickler zweiter Klasse abgestempelt zu werden.

Eine interessante Analogie hierzu ist die Dombauhütte Köln, deren 60 Mitarbeiter den Kölner Dom in Stand halten. Keiner dieser Handwerker wird seine Aufgabe beschreiben mit „Es ist furchtbar, diese alte Kirche mit alter Bauweise und altem Material am Leben zu erhalten". Im Gegenteil: Jeder wird mit stolzgeschwellter Brust davon berichten, dass er ein Teil der Gruppe ist, die das Weiterbestehen dieses Teil des Weltkulturerbes sicherstellt.

Nun ist es sicherlich ein wenig weit hergeholt, ein Stück Software mit einer Hunderte Jahren alten Kirche vergleichen zu wollen, aber dennoch: Auch Entwickler, die sich mit „langweiliger" Legacy-Entwicklung beschäftigen, sind ein essenzieller und wichtiger Bestandteil unserer Entwickler-Community, die stolz auf ihre Arbeit sein können (und sollten).

Karriere-Management

Als Softwareentwickler liegt unser Fokus selten auf einer Karriere im klassischen Sinne: dem Sammeln von Privilegien und dem Erklimmen immer neuer Stufen in der firmeninternen Hierarchie. Dennoch lohnt es sich, hin und wieder innezuhalten und zu überlegen: Wo stehe ich im Moment? Was sind meine Optionen und möglichen nächsten Schritte? Was sind Dinge, die ich bisher gelernt habe? Worauf möchte ich mich in Zukunft konzentrieren und fokussieren? Welche Pfade stehen mir offen und welche davon möchte ich bewusst und aktiv wählen?

Eine Beobachtung, die man hier häufig machen kann, ist: Wer seine Karriere nicht aktiv gestaltet, für den wird sie von jemand anderem gestaltet. In vielen Köpfen existiert immer noch „der" klassische Weg eines Softwareentwicklers, der nach einigen Jahren vorsieht, weniger Code zu schreiben und dafür mehr organisatorische Aufgaben zu übernehmen. Während für manch einen genau das ein wünschenswerter Entwicklungspfad ist, so finden sich auch immer wieder Beispiele von vormals exzellenten Technikern, die nach einer Beförderung auf eine Management-Position weder glücklicher noch produktiver waren als vorher. Die harte Wahrheit ist auch hier: Nicht jeder Softwareentwickler ist automatisch ein guter Manager.

Es ist daher mehr als hilfreich, ein realistisches Verständnis davon zu haben, was die eigenen Stärken, aber auch die eigenen Schwächen sind. Karriereschritte sollten nicht danach ausgewählt werden, was im Lebenslauf am besten aussieht oder welche Zusatzleistungen damit verbunden sind, sondern vielmehr danach, ob sie den eigenen Interessen entsprechen und interessante und vor allem gewollte Herausforderungen mit sich bringen.

Continuous Improvement

In agilen Entwicklungsteams ist die regelmäßige Retrospektive ein wichtiger Bestandteil von kontinuierlicher Verbesserung. Was sind Dinge, die gut liefen? Was sind Dinge, die verbessert werden sollten? Welche kleinen Schritte können wir ab morgen unternehmen, um noch besser zu werden?

Es lohnt sich, diese Denkweise auch für die eigene Arbeitsweise anzusetzen und sich selbst in den Mittelpunkt zu stellen. Was sind Dinge, die in der letzten Zeit gut gelaufen sind? Was sind Dinge, die mir besonders Spaß gemacht haben und/oder die ich neu gelernt habe? Was davon lohnt es sich, weiterzuverfolgen und was davon hat sich als nicht interessant herausgestellt?

Große und radikale Änderungen umsetzen ist immer schwierig und aufwendig – das gilt für Softwareprojekte ebenso wie für eigene Verhaltensweise. Agile Entwicklung hat uns „Inspect and adapt" gelehrt: Viele kleine Anpassungen lassen sich deutlich einfacher umsetzen.

Der Blick nach vorne

Die Schnelllebigkeit der Softwareindustrie birgt sowohl Chancen als auch Risiken. Chancen dadurch, dass sich der eigene Aufgabenbereich und die eigene Expertise innerhalb weniger Jahre grundlegend ändern kann. Risiken dadurch, dass man häufig kaum dazu kommt, innezuhalten und zu reflektieren, ob das, was man gerade macht, für einen selbst noch interessant und sinnstiftend ist. Wer sich aber immer wieder bewusst dazu entscheidet, einen offenen und ehrlichen Blick auf sich selbst zu riskieren, der wird auch lernen, sicher durch diese rauen Gewässer zu navigieren.

Christian Seifert
BetterDoc GmbH
christian.seifert@betterdoc.de

Christian Seifert ist Software Engineer mit fast 20 Jahren Erfahrung in der Entwicklung und dem Support von Individualsoftware. Ob beim Schreiben von Code, der Analyse von Anforderungen oder dem Mentoring im Team – für ihn geht es immer darum, genau die Lösung zu finden, die auch tatsächlich von Anwendern benötigt und gewollt ist. Auch wenn ihn ursprünglich die Faszination an der Technik zur Softwareentwicklung trieb, so verbringt er heute mindestens genauso viel Zeit mit Menschen und wirbt leidenschaftlich für ein besseres Verständnis zwischen Entwicklungsteams und anderen Stakeholdern.