Unsere neue „Aus der Szene“ -Interview-Marke

Szene-Trends nachgefragt – in dieser Ausgabe bei Dirk M. Sohn

„Softwareentwickler sind vom Naturell her Ingenieure: Gar nicht so sehr am Geld interessiert, sondern am Sinn“

IT-Berater Dirk M. Sohn über Nachwuchssorgen und die Attraktivität moderner „Softwareentwicklungs-Unternehmen mit fachlichem Wurmfortsatz“ – Banken und Speditionsunternehmen

Der 47-Jährige ist Experte im Bereich Individual-Software und Entwicklung mit Java EE. Der Geschäftsführer von „OIO Orientation in Objects GmbH“ hat langjährige Erfahrung als Berater für Agile Methoden und Softwareentwicklungsprozesse aus Management-Sicht.
Der Mannheimer Informatiker macht sich Gedanken über Verschwendung: Vor allem über die Verschwendung von Kreativität, guten Mitarbeitern und vielversprechenden Ansätzen in einer Zeit, in der ständig IT-Neuentwicklungen gebraucht werden. JavaSPEKTRUM hakte nach.

JavaSPEKTRUM: *Software ist überall. Immer mehr Unternehmen sind auf gute Entwickler angewiesen. Wie kommen Firmen an geeignete Informatiker, was muss sich in den Betrieben ändern?*

Dirk M. Sohn: Unternehmen müssen sich Softwareentwicklung mehr als kontinuierlichen Fluss vorstellen, anstatt in Projekten zu denken. Nicht nur in dem einen Projekt leben, das leider zum Scheitern verurteilt ist, weil zum Beispiel der Termin zu kurz gesetzt ist. Mit klassischem Projektmanagement wurde und wird teilweise auch heute noch Softwareentwicklern beigebracht, in Einzelprojekten zu denken. Das war und ist notwendig für die Entwicklung von Raketen oder einer Atombombe, im Brückenbau usw. Im Bereich der Softwareentwicklung werden aber heute fast durchgängig immer wieder kurzfristig neue Funktionen auf der Basis eines komplexen bestehenden Systemumfeldes gebraucht. Deshalb ist es hier wichtig, den Projektbegriff aufzulösen. Ich favorisiere kleine Entwicklergruppen, die selbst gelernt haben, wie sie als Entwickler in ihrem Kontext am besten arbeiten. Die kontinuierlich Software zum Kunden bringen. Fließend. Qualität und Werte fließen auf diese Art weit mehr in die Arbeit ein als bei einer „ruckartig“ projektorientierten Vorgehensweise.

Das funktioniert?

Es ist ein Anfang, mit diesem Gedanken zu spielen. Die Umsetzung ist nicht so leicht in einer größeren betrieblichen Organisation. Da gibt es klassischerweise ein Projektmanagement-Office, Projektmanagement-Schulungen und Projektleiter. Man kann nicht von heute auf morgen alles schließen, hingehen und sagen: „Wir brauchen sie alle nicht mehr“.

Sie arbeiten mit Schablonen?

Blueprints nennt sich das eher. Wir kommen mit einer Vorstellung zur Art und Weise der Zusammenarbeit in großen Teams, wie beispielsweise dem Scaled Agile Framework (SAFe), und schauen, ob sie zum Kunden passt. Verschiedene Phänomene werden häufig über verschiedene Kunden hinweg gleich sein. Warum soll man ihnen nicht industrieweit gleiche Namen auf einem Blueprint geben dürfen? Wenn man beispielsweise sechs Entwicklungsteams mit Arbeit versorgen muss, die alle am gleichen Wertstrom oder an der gleichen Lösung arbeiten, ist es gut, wenn man Leute hat, die das alles fachlich überblicken. Wenn man die Leute, die diesen fachlichen Überblick gewährleisten, einfach mal Product-Management nennt, macht man nicht viel falsch. Daneben gibt es diejenigen Leute, die in den verschiedenen Entwicklungssträngen aufpassen, dass technische Wiederverwendung unter Umständen Kosten spart oder einheitliche und somit kommunizierbare Lösungsansätze schafft. Warum soll man die nicht Softwarearchitekten nennen? Das Wort gibt es ja. Ich fand es spannend, dass in der „Ursprungs-agilen Szene“ alles heilig und sakral war. Das war für eine Weile auch nötig. Man durfte zum Beispiel nicht von Management-Werkzeugen für Aufgabenplanung sprechen. Solche Werkzeuge waren böse. Alles musste haptisch auf Aufgaben-Kärtchen gemacht werden. Man durfte auch nicht von Kapazitätsplanung sprechen. Volles Commitment und Vollzeit-Einsatz waren eh vorausgesetzt, warum hätte man über Einsatzplanung reden sollen? Genauso sollte man eigentlich keine vorauseilende Architektur entwickeln. Die Teams sollten die Architektur aus der Arbeit und dem Bedarf heraus erarbeiten. Diese Gedanken waren im Prinzip zunächst richtig und wichtig, umso mehr, solange es sich nur um kleine Teams handelte und viele falsche Ideen der Vergangenheit überwunden werden mussten. Aber wenn man sechs agile Teams losschickt und schaut, ob die nicht doch gemeinsam eine komplexe Software bauen können, muss man eine Instanz haben dürfen, die die Architektur-Themen überblickt. Und diese sollte Architekt heißen und vordenken dürfen.

Was kommt heraus, wenn ganz viele solcher Teams auf möglichst agile Art arbeiten?

Erfolgreiche Entwicklungsfortschritte. Viele Management-Vorträge zitieren die Standish Group und ihren jährlich erscheinend Chaos-Report, demzufolge immer noch 60 Prozent aller Projekte scheitern oder zumindest nicht gelingen. Diese Zahl bedeutet unheimlich viel Geld, menschliche Seele und Energie, die auf der Straße bleiben, nur weil dauernd Softwareentwicklung scheitert. Software ist in immer mehr Geräten und in immer mehr Zeit unserer Lebenswirklichkeit drin. Die modernsten unserer Kunden beginnen, sich als Softwareentwicklungs-Unternehmen mit fachlichem Wurmfortsatz zu begreifen, obwohl sie bis vor Kurzem noch eine Bank oder ein Speditionsunternehmen waren. Das sind heute eigentlich Programmierer mit Geldausgabeautomaten bzw. Lastwagen. Wenn diese Kunden dauernd scheiternde Projekte hätten, dann verlören sie ihre Programmierer, hin zu besseren Arbeitgebern, bei denen die Softwareentwicklung nicht scheitert. Das ist dann ein ganz großes Problem. Das sind nicht nur zehn Prozent weniger Aktionärsrendite.

Softwareentwickler sind gefragte Leute?

Sie werden immer gefragtere Menschen. Wir sind eine aussterbende Gesellschaft. Die Debatte über Zuwanderung ist nicht abgeschlossen. Ich glaube, dass gut eingespielte, kleine Teams in der Softwareentwicklung unheimlich viel bewirken und Sinn stiften. Das ist der Wettbewerbsvorteil von Arbeitgebern, die solche Stellen anbieten können.

Wären Spaß und Freiheit Argumente, Softwareentwickler zu binden?

Softwareentwickler sind vom Naturell her eher Ingenieure. Sie sind gar nicht so sehr an Geld oder an Spaß interessiert in der täglichen Arbeit, sondern an Sinn. Ein gut eingespieltes kleines Team kann zehnmal produktiver sein als ein großer Haufen, der nicht funktioniert – und das macht Sinn. Teams von fünf oder sechs Leuten haben Projekte gestemmt, die an anderen Stellen mit sechzig Beteiligten nicht geklappt haben.

Wie kann diese Entwicklung angeschoben werden?

Blueprints sind bestimmt kein Allheilmittel. Sie sind aber eine gute Orientierung, um einen großen Schritt nach vorne zu kommen. Mehr im Sinne von: Irgendwie so könnten wir zusammenarbeiten, lasst uns das ausprobieren und lernen. Das ist eine jahrelange Lernzeit, normalerweise.

Sind Smart Home und Internet of Things Anschieber?

Sicherlich sind diese beiden Themen Treiber einer kontinuierlichen Software-Auslieferung. Der Bedarf fortlaufend gelieferter Software entsteht durch entsprechende Software-Abnehmer. Viele auch prominente Software-Produzenten wie zum Beispiel SAP, Microsoft oder IBM müssen ihre Software aktuell allerdings noch mit manuellen Eingriffen an ihre Abnehmer ausliefern. Sie schaffen es nicht, den Kunden diese Rollouts alle zwei Wochen zuzumuten. Es ist auch nicht realistisch, alle zwei Wochen neues Schulungsmaterial zu drucken. Sich trotzdem einer fortlaufenden Software-Auslieferung zumindest als Möglichkeit anzuschließen, versetzte aber auch diese Produzenten zumindest in die Lage, jedem einzelnen Kunden zu jeder Zeit die neuesten Funktionen anbieten zu können.

Wie lässt sich das befördern?

Das Umfeld der Softwareentwicklung muss sich darauf einstellen und die entscheidenden Fragen stellen: Wo bekomme ich kontinuierlich neue Features her, wo neue Ideen für die Software? Wie funktioniert mein Innovationsprozess? Das Controlling muss sich ändern. Man kann Scheinpläne aufstellen, die dann doch schiefgehen. Und Scheinbudgets. Das bringt aber alles nichts. Man sollte eher tatsächliche Messwerte entnehmen, mit denen man dann feinsteuert und lernt, also empirische Prozesse bevorzugen. Die ganze Betriebsorganisation muss entsprechend mit umdenken. Irgendwann wird das ein Thema in der kompletten Organisation, nicht nur in der Softwareentwicklung.

Wie würden Sie einem ganz klassisch aufgebauten Unternehmen erläutern, was sich ändern muss?

„Ihr müsst ein kleines Team schaffen, das gerne und gut Software entwickelt. Das ist der Anfang. Das kann sehr viel Weiteres von alleine und mit der Hilfe von Blueprints aufbauen. Und ihr müsst zuvor dieses Team, falls nicht schon vorhanden, auf einen bestimmten Ausbildungsstand heben." – Ein Problem an diesem selbstverantwortlichen Arbeiten, beispielsweise in einem Scrum-Team, ist: Es geht von einem eigenverantwortlich handelnden Menschen aus, der eine hohe Bildung mitbringt. Dummerweise ist das mehrheitlich so gar nicht gegeben. Unternehmen, deren Kerngeschäft nicht Softwareentwicklung ist, haben es sehr schwer, gute Programmierer zu finden, weil die gar keine Lust haben, in einem Umfeld zu arbeiten, in dem exzellente Softwareentwicklung kein anerkannter Erfolgsfaktor ist. Tolle Programmierer gehen vielleicht zu SAP, aber typischerweise nicht zu einer regionalen Bank. Die Lösung: Man muss die Menschen fit machen. Der erste Schritt ist handelnde Weiterbildung. Keiner kann nur mal schnell auf eine Schulung gehen, um ein guter, selbstverantwortlicher, teamfähiger Programmierer zu werden. Man braucht ein reales Vorhaben, das nicht so kritisch ist. So, dass das Team bei seinen ersten Schritten gleich ein paar, aber nicht zu viele Schmerzen verursacht, wenn es irgendwo scheitert. Jeder lernt aus Fehlern. Ein Vorhaben, ein Team, ein bisschen Ausbildung – in einem kleinen Kontext kann das sehr schnell klappen. Nach einem Jahr, ein paar Auslieferungen, kann es sein, dass es dann richtig Spaß macht.

Ist das ein Weg, Menschen an den Beruf und in Unternehmen zu bringen, die auf den ersten Blick gar nicht so attraktiv wirken?

Als Aktionär würde ich die Aktien derer kaufen, die das schaffen. Ich glaube, dass es unheimlich wichtig für den wirtschaftlichen Erfolg im 21. Jahrhundert ist, dass die Betriebe lernen, gute Softwareentwickler zu beschäftigen. Konventionelle Automobil-Hersteller werden beispielsweise auch von solchen Zukunftssorgen geplagt. Sie glauben nicht, Tesla würde zu viele Elektroautos ausliefern, sondern Google zu viele selbstfahrende Autos produzieren. Womit fahren die Autos aber selbst? Mit Software. Die Haushaltsautomatisierung macht jetzt tatsächlich die Fortschritte, dass sie im Markt ankommt. In immer mehr Branchen steht und fällt der wirtschaftliche Erfolg mit der Softwareentwicklungskompetenz.

„Einer, der gut werden will, muss irgendwann sein erstes eigenes Problem lösen. Je früher, desto besser."

Wie kann man schon Schülern Lust auf eine Karriere als Entwickler machen?

Bei uns damals fand ich es logisch, dass wir fitter waren als die Lehrer. Damals war die Materie noch ganz neu. Aber auch heute schafft man es nicht, ausreichend viele fitte Programmierer als Lehrer an die Schulen und Berufsschulen zu bekommen – das muss man sich eingestehen und reagieren. Dem gewerblichen Informatik-Azubi würde eine ordentliche schulische Ausbildung zustehen. Der Kaufmann bekommt eine ganz gute Ausbildung an der Berufsschule, der Informatiker aber nicht. Ein weiteres Thema ist: „Frauen in die IT". Es ist ein hochinteressanter Beruf für alle Talente, die man „weiblich" nennt. Die können natürlich auch Männer haben. Ich meine speziell, die menschliche Kommunikation mit logischem Sachverstand zu verbinden. Der Girls' Day ist hier ein nettes Beispiel für eine kleine Initiative. Wieso das bei mir damals ganz klar war, Anfang 1983 mit 15 meine Eisenbahnanlage zu verkaufen und mir von dem Geld einen Commodore 64 zuzulegen, weiß ich nicht mehr. In jedem Fall hatte ich aber zuvor schon einen Lötkolben in der Hand. Das ist heute nicht mehr typisch. Kinder reparieren und konstruieren immer seltener, ist mein Eindruck. Heute, wo der Alltag zunehmend weniger Anreize schafft, muss man vermutlich einen Weg finden, wie begeisterte Programmierer Kindern früh das Entwickeln-Erleben nahebringen. Hochschulen vermitteln sicherlich auch spät noch gute theoretische Grundlagen. Eine Einstellungstestfrage an jeden Bewerber ist bei uns grundsätzlich: „Was hast du denn selbst schon programmiert?" Häufige Antwort: „Eigentlich nur die Übungen an der Hochschule." Das reicht aber bei Weitem nicht. Einer, der gut werden will, muss irgendwann sein erstes eigenes Problem lösen. Je früher, desto besser. Damit das Thema „Softwareentwicklung" Schule machen und für Nachwuchs sorgen kann, müsste das Thema „Softwareentwicklung-Erleben" schon frühzeitig an die Schulen gebracht werden.

Interview: Annegret Handel-Kempf