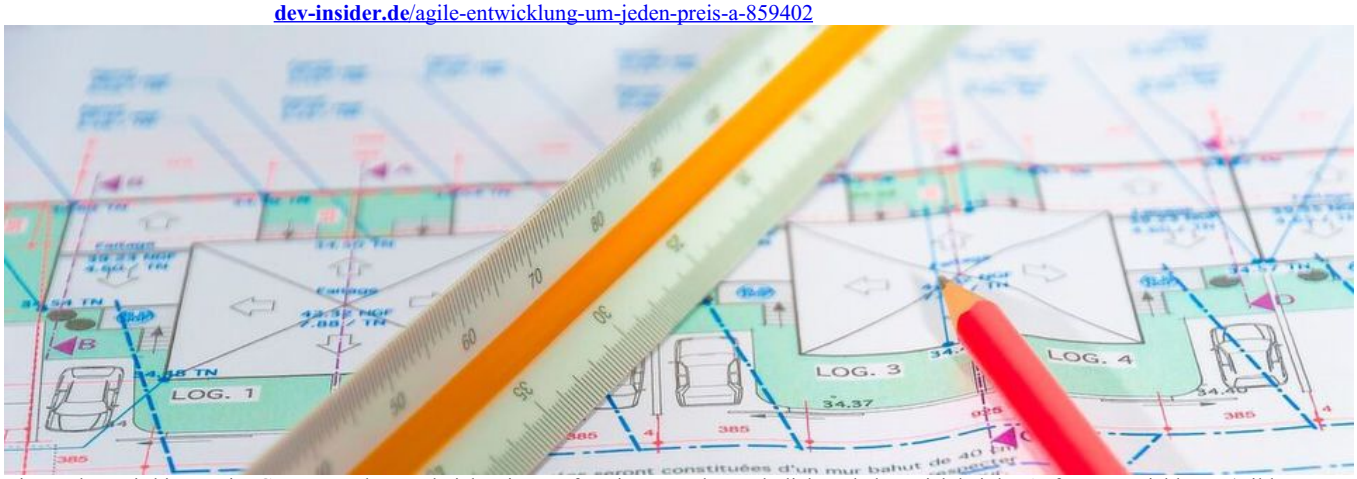
Agile Entwicklung um jeden Preis?

dev-insider.de/agile-entwicklung-um-jeden-preis-a-859402



Ein Neubau wird immer im Ganzen geplant und nicht Zimmer für Zimmer gebaut, ähnlich verhält es sich bei der Auftragsentwicklung. (Bild gemeinfrei: caropat / Pixabay)

Die direkte Zusammenarbeit von Softwareentwicklern und Anwendern ist das, was moderne IT-Projekte auszeichnet. Doch geht es auch ohne Sprints und Projekt Owner?

Heute herrscht weitgehend ein Konsens darüber, dass Softwareentwicklung nach dem Wasserfallmodell faktisch nicht möglich ist. Die wenigen noch verbliebenen Anhänger von Wasserfall, V-Modell und V-Modell XT wirken heute wie aus der Zeit gefallen.

Einigkeit scheint es zudem darin zu geben, dass Agile Softwareentwicklung heute das Maß aller Dinge sei. Jedoch haben nicht alle IT-Verantwortlichen durchweg positive Erfahrungen mit agiler Softwareentwicklung gemacht.

Von einem wirklichen Einvernehmen, wie man bei der Entwicklung kundenspezifischer Softwarelösungen vorgehen sollte, sind wir heute weiter entfernt denn je. Das hat damit zu tun, dass Agile Softwareentwicklung nach Scrum tatsächlich nicht immer und nicht überall der richtige Weg ist.

Warum Scrum nicht grundsätzlich die beste Wahl ist

Man sollte sich immer vergegenwärtigen, vor welchem Hintergrund Scrum in den 90ern erfunden wurde: als die optimale Organisationsform für eine eigene, begrenzte Entwicklermannschaft für ein eigenes Produkt oder Online-Angebot.

Wenn „Time to Market“ das wichtigste Anliegen ist, dann ist es sinnvoll, so schnell wie möglich eine erste Version auf den Markt zu bringen bzw. live zu gehen und dann mit dem vorhandenen Personal in vergleichsweise kurzen Zyklen immer wieder neue, verbesserte Versionen nachzulegen. Und das immer so weiter – Softwarepflege bis in alle Ewigkeit.

Aus dem natürlichen, jahrelangen Zusammenspiel der Eigentümer des immer weiterentwickelten Produkts mit festen Entwicklerteams resultieren die Scrum-Rollen des Product Owners mit seinem „Backlog“ und des Scrum Managers als Koordinator vieler eingespielter und jeweils relativ selbstständig agierender Teammitglieder. Leider passt aber all dies nicht so recht auf den Fall einer typischen Auftragsentwicklung – bzw. wenn überhaupt, dann erst in der späteren Pflegephase. Warum?

Bei einer Auftragsentwicklung will man so schnell wie möglich und so kostengünstig wie möglich eine fertige Software geliefert bekommen, die alles enthält, was bestellt wurde. Zwischenlieferungen sind nur Zeit- und Geldverschwendung. Es ist zwar richtig, dass die meisten Auftraggeber gern Zwischenstände sehen möchten, um neue Erkenntnisse in die Entwicklung einbringen zu können und um sicher zu sein, dass man auf dem richtigen Weg ist. Aber jeden Monat eine voll getestete, aber nutzlose Version abnehmen zu müssen, das ist sicherlich nicht im Interesse des Auftraggebers.

Bei einer Auftragsentwicklung muss man normalerweise immer mit einem fixen Budget auskommen. Deshalb mögen Auftraggeber Festpreisprojekte. Eine Scrum-Entwicklung aber ist quasi ein Blanko-Scheck für den Auftragnehmer. Selbst wenn jeder einzelne Sprint als separater Festpreis vereinbart wird: Für den Auftraggeber wird das oft zu einer endlosen Folge von immer wieder neuen Versionen mit immer wieder neuem Gang zum Haushälter.

Fast immer mangelt es an einem kompetenten Product Owner, der alles im Blick hat, der das gesamte Projekt versteht, der sämtliche Anforderungen der Anwender aufnehmen und in die IT-Sprache übersetzen und zudem auch noch diese unendlich vielen Wünsche vernünftig priorisieren kann.

Für den Gründer eines Startups mag das für sein Projekt noch klappen, aber in einer großen Organisation wird man kaum jemanden finden, der diesem Anspruch wirklich gerecht wird. Wenn aber jemand zwischen Anwender und Entwickler steht, der von allen Beteiligten am wenigsten davon versteht, worum es eigentlich geht, dann ist dieser Ansatz von Vornherein zum Scheitern verurteilt.

Viele IT-Chefs berichten, dass Scrum-Projekte, sofern sie denn überhaupt jemals fertig geworden sind, mindestens das Doppelte bis Dreifache

gekostet und doppelt bis dreimal so lange gedauert haben, als wenn man sie zum Festpreis vergeben hätte. Wer sich diesen Overhead leisten kann, für den mag Scrum durchaus sinnvoll sein, um das Produkt gut an der Realität ausreifen zu lassen.

Und es gibt noch einen weiteren, einen technischen Grund, der es nahelegt, Scrum für Auftragsentwicklungen zumindest kritisch zu betrachten. Und zwar sind die meisten ausgeschriebenen Business-Anwendungen im Kern Datenbankanwendungen. Es liegt in der Natur der Sache, dass Datenbankanwendungen so aufgebaut werden müssen, dass das zugrundeliegende Datenmodell gewissermaßen das Fundament der weiteren Entwicklung ist.

So wie man kaum ein Haus nach Scrum entwickeln kann, indem man ein Zimmer nach dem anderen plant und dann jeweils das Fundament um ein paar Meter verlängert, so kann man auch nicht ein Feature nach dem anderen einbauen, ohne dabei das Datenmodell dermaßen zu beeinträchtigen, dass das ganze Projekt daran scheitern wird. Ein ständiges Umbauen des Datenmodells, ganz typisch für Scrum-Entwicklungen, führt zu teilweise absurden Mehraufwänden und/oder zu einer schlechten Architektur der Anwendung.

Alternative für Auftragsentwicklungen

Um dieses Problem zu lösen und dennoch ein Maximum an Agilität zu ermöglichen, wurde das Phasenagile Vorgehensmodell entwickelt. Da wird das Haus nicht von links nach rechts oder von innen nach außen gebaut, sondern so wie es eigentlich üblich ist: Von unten nach oben. Genau dieses Prinzip gilt für die moderne Softwareentwicklung:

Phase 1 betrachtet das Gesamtprojekt, also auch all das, was erst einmal nur im Backlog steht und vielleicht eines Tages noch beauftragt wird. Nur so ist es möglich, die Grundarchitektur und das Datenmodell des Systems auf eine hinreichend solide Basis zu stellen.

Anschließend wird in Phase 2 das Fundament geplant und gegossen. Das heißt, dass alle Entitäten des künftigen Gesamtsystems bereits mitberücksichtigt und zueinander in die richtigen Beziehungen gesetzt werden, aber auch alle Schnittstellen in die Außenwelt und andere technische Grundaspekte sind mit einbezogen. Das heißt nicht, dass man schon alle Datenbanktabellen final attributieren müsse, denn es geht eher um das große Ganze.

Spätestens hier stellt sich die Frage, wie man es schaffen kann, beim Datenmodellieren bereits die künftigen Anwender mit einzubeziehen, wo die doch angeblich nur in Prozessen, aber nicht in Datenstrukturen denken können. Aber weit gefehlt: Man muss nur wissen, wie man das macht. Ganz sicher nicht, indem man mit Anwendern ER-Modelle an der Tafel diskutiert. Man muss mit geeigneten technischen Mitteln die Datenstrukturen „lebendig“ und verständlich machen – anhand echter Kunden- oder realitätsnaher Testdaten.

Danach geht man weiter, von Etage zu Etage, von Phase zu Phase, und es wird immer agiler. Bei aller Freiheit, die jeweiligen Details immer weiter auszuprägen, geht dies stets immer nur in genau der jeweiligen Phase. Im Anschluss wird sozusagen die Tür zugemacht und spätere Änderungen sind nur noch in Ausnahmefällen zulässig.

Wie viele Phasen ein Projekt haben sollte, hängt von den verwendeten Softwaretechnologien, dem Charakter des Projekts und vielen anderen Randbedingungen ab.

Phasenagiles Vorgehen ist optimal für Low-Code-Entwicklung

Das Phasenagile Vorgehensmodell ist zwar nicht an bestimmte Entwicklungsmethoden gebunden, aber optimal für die Arbeit mit Rapid-Development-Werkzeugen und Low-Code-Entwicklungsumgebungen. Low-Code-Plattformen erlauben eine Softwareentwicklung ohne oder fast ohne Programmierung, lediglich durch interaktive Konfiguration in einer Art „Cockpit für Business Developer“. Dies verleiht den Entwicklern nicht nur eine ungeahnte Entwicklungsgeschwindigkeit, es ist auch hervorragend dazu geeignet, die künftigen Anwender direkt in den Entwicklungsprozess mit einzubeziehen.

Das heißt natürlich nicht, dass man zwingend vor Ort entwickeln muss, oder dass die Anwender permanent den Entwicklern über die Schulter schauen würden. Eine Möglichkeit wären wöchentliche Workshops mit Besprechungen und Sofortoptimierungen der laufenden Arbeitsstände direkt am Bildschirm, und zwar ganz bewusst im direkten Dialog direkt zwischen Entwicklern und Anwendern.

Für die Durchführung der Arbeitsworkshops empfehlen sich die Prinzipien des „Design Thinking“. Leitlinien des „Design Thinking in allen Projektphasen“ sind die Werkzeuge und Regulatoren dieser Art der Zusammenarbeit in interdisziplinären Teams. Ganz wichtig dabei ist, dass die Projektleiter beider Seiten (Auftragnehmer und Auftraggeber, bzw. Fach- und IT-Abteilung) mehr Moderatoren als Owner der zu entwickelnden Produkte und Projekte sind.

Wenn dies erfolgreich umgesetzt wird, dann eröffnet es den Anwendern ein deutliches Mehr an Einflussnahme-Möglichkeiten, als es bei einer Scrum-agilen Entwicklung jemals möglich wäre. Der Anwender ist viel dichter an den eigentlichen Entwurfsprozessen dran und neue Ideen lassen sich viel schneller, teils sofort, auf ihre Umsetzbarkeit und Tauglichkeit prüfen. Wenn es das Budget erlaubt, gerne auch mit Giebeln und Türmchen, ansonsten eher zweckmäßig nüchtern.

Karsten Noack (Bild: berlin-event-foto.de)

Aber wie auch immer – ob mit oder ohne Pflichtenheft, ob Festpreis oder nach Aufwand, ob mit oder ohne Giebel und Türmchen: Ein gutes und kooperatives Projektmanagement ist bei allem stets das Entscheidende.

** Karsten Noack ist Gründer und CEO der Scopeland Technology GmbH. Als Visionär entwickelte er bereits Mitte der 90er Jahre die Grundlagen der Technologie, die heute als Low-Code und als Schlüsseltechnologie der Digitalisierung bekannt ist. Er ist Mitglied des Hauptvorstands des BITKOM sowie mehrerer Arbeitsgruppen des IT-Branchenverbands. Als Geschäftsführer und Cheftechnologe von Scopeland Technology trägt er die Hauptverantwortung für die strategische Ausrichtung und ist in die wichtigsten Projekte des Unternehmens involviert.*