Im Angesicht der Unvorhersehbarkeit

Die **zentrale Kraft** der Nachhaltigkeit

Investitionssicherheit ist eine fundamentale, seltsamerweise jedoch meist unterspezifizierte Anforderung an Software. Ihre Erfüllung kann nicht durch Messungen am Code festgestellt werden. Also muss der gelebte Softwareproduktionsprozess sie sichern.

Nachhaltigkeit ist nur erreichbar, wenn Unvorhergesehenes nicht als zu vermeidender Sonderfall angesehen wird, sondern als erwünschte Regel. Nachhaltigkeit ist eine Anforderung an Software beziehungsweise den Softwareproduktionsprozess. Anders als bei Anforderungen in den Kategorien Funktionalität und Effizienz lässt sich Nachhaltigkeit jedoch nicht so einfach durch den Auftraggeber „am Objekt" überprüfen. Codemetriken – zum Beispiel Zyklomatische Komplexität, Anzahl Dubletten – mögen Hinweise darauf geben, wie sich die Nachhaltigkeit entwickelt.

Letztlich jedoch ist Nachhaltigkeit eine Eigenschaft des sozio-technischen Systems bestehend aus Team und Code/Infrastruktur. Und ob sie hoch oder niedrig ist, wie sie sich tendenziell entwickelt, das ist eine Frage des Flusses.

Die Realität

Ist die Softwareproduktion nachhaltig, dann empfindet der Auftraggeber die Übersetzung von Anforderungen in lauffähige Software als flüssig. Dem steten Strom neuer Anforderungen entspricht ein steter Strom von Code-Inkrementen, die sie erfüllen. Der hat etwas von Vorhersehbarkeit, ohne deshalb aber berechenbar zu sein. Es geht vielmehr um die Anwesenheit von Verlässlichkeit und die Abwesenheit von bösen Überraschungen.

Böse Überraschungen für den Auftraggeber sind unerwartete Sprünge in der Durchlaufzeit von Anforderungen. Dass größere Anforderungen länger als kleinere in der Umsetzung dauern, ist ja selbstverständlich. Doch dieser Zusammenhang sollte im Wesentlichen linear sein.

In der Realität jedoch, solange nicht auf Nachhaltigkeit, das heißt stete Wandelbarkeit von Software Wert gelegt wird, kommt es häufig zu Brüchen und Stockungen im Softwareproduktionsfluss. Dann springt die Umsetzungszeit plötzlich exponenziell nach oben.

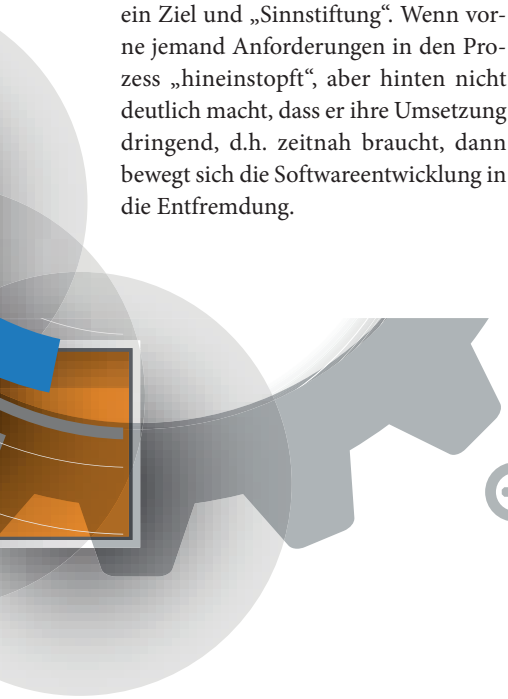
An dieser Stelle seien zwei typische, tiefsitzende Missverständnisse dafür beispielhaft als Ursache genannt:

1. Man meint, es sei wichtig, die Softwareproduktion mit Anforderungen zu versorgen, damit es nicht zu Leerlauf kommt.
2. Die Anforderungen an die Softwareproduktion meint man umfassend und lückenlos liefern zu müssen, damit zügig produziert wird.

Natürlich muss die Softwareproduktion mit Anforderungen versorgt werden – doch der Fokus auf die Anforderungsdefinition führt zu einem blinden Fleck. In dem liegt die Abnahme der Umsetzung von Anforderungen. Dieser blinde Fleck zeigt sich in der Literatur, die vornehmlich über Requirements Engineering, die Formulierung von User Stories und Use Cases sowie Planungsmeetings vor der Umsetzung schreibt. Aber was passiert nach der Umsetzung? Da soll die Qualitätssicherung einfach nur fest-

stellen, dass alles auch wie gewünscht umgesetzt wurde.

Wie sich jedoch zeigt, ist solches Vorgehen kraftlos. Ihm fehlt nicht der Druck (push), sondern der Sog (pull). Es fehlt Kohärenz, es fehlt Führung im Sinne koordinierender Ausrichtung auf ein Ziel und „Sinnstiftung". Wenn vorne jemand Anforderungen in den Prozess „hineinstopft", aber hinten nicht deutlich macht, dass er ihre Umsetzung dringend, d.h. zeitnah braucht, dann bewegt sich die Softwareentwicklung in die Entfremdung.



Die wichtigste Kraft

Die wichtigste Kraft in der Softwareproduktion ist die Sog- oder Zugkraft am Ende des Herstellungsprozesses. Eine Lieferung des Ganzen in mehreren Monaten ist dafür nicht ausreichend. Nein, es geht um die tägliche (!) Präsenz desjenigen, der die Software haben will. Das ist der Kunde bzw. sein Stellvertreter, der Product Owner. Der muss jeden Tag „auf der Baustelle sein" und am Entwicklungsteam ziehen; verlässlich, d.h. mit Ansage: „Morgen komme ich wieder und möchte X sehen."

Es geht dabei nicht um die Formulierung von Anforderungen, es geht auch nicht darum, für Fragen zur Verfügung zu stehen. Nein, der Product Owner soll fordernd präsent sein. Nur indem er ein tägliches „Umsetzungsvakuum" herstellt, d.h. eine Leerstelle, die durch Lieferung eines weiteren Softwareinkrements gefüllt werden muss, macht er deutlich, dass ihm an Produktionsergebnissen gelegen ist.

Solche Abnahme ist die vornehmste Aufgabe des Product Owners. Sie rangiert weit vor der Anforderungsdefinition. Denn nur eine solch ständige Abnahme erzeugt einen Zug an der Programmierung, dass sie sich – mit welchen Maßnahmen auch immer – auf Wandelbarkeit einstellt. Nur wer täglich in der Pflicht steht, etwas zu liefern, empfindet Notwendigkeit, kontinuierlich zu testen oder Prinzipien des Clean Code Development (www.clean-code-developer.de) anzuwenden.

„Dreh- und Angelpunkt nachhaltiger Softwareproduktion ist ein starker, klarer Abnehmer, der die Programmierung eng begleitet."

Ralf Westphal, freiberuflicher Berater, Projektbegleiter und Trainer für Themen rund um Softwarearchitektur und Teamorganisation

Der Product Owner ist mithin die wichtigste Rolle in der Softwareproduktion [1]. Ohne einen zugkräftigen, präsenten, entscheidungsmächtigen Product Owner kommt Softwareentwicklung ins Stocken. Sie verstrickt sich in Details, verirrt sich in Technologien, vergoldet Erker und Türmchen, statt flüssig das Nützlichste zu liefern.

Solche Wichtigkeit der Rolle erzwingt natürlich, dass sie exklusiv besetzt wird. Product Owner mit anderen Aufgaben zu überladen, darf keine Option sein. Die schwächsten, die teuersten Projekte sind die, bei denen Product Owner gleichzeitig noch Projektleiter, Produktmanager, gar Geschäftsführer sind.

Ungenauigkeiten reduzieren

Leider ist das jedoch die wenig nachhaltige Realität, die dem zweiten Missverständnis Vorschub leistet. Weil Product Owner überladen mit anderen Aufgaben sind, wollen sie sich die Zeit für den ständigen Zug sparen und meinen, sie könnten sich mit umfassenden Anforderungen „freikaufen". Das funktioniert jedoch nicht.

Zum einen wächst die Ungenauigkeit überproportional mit dem Umfang der Anforderungen, die auf einmal in die Softwareproduktion „gekippt" werden. Das führt dann zu schlechterer Qualität des Outputs der Softwareproduktion und kapazitätsverringernden Nachbesserungen.

Zum anderen suggerieren umfassende und vermeintlich lückenlose Anforderungen eine Vorhersehbarkeit, die letztlich nicht vorhanden ist. Ein weiter Anforderungshorizont führt zu starren Strukturen in der Software, die das sichtbare und vermeintlich Stabile innerhalb dieses Horizonts optimiert bedienen sollen.

Fazit

Angesichts der grundsätzlichen Natur von Anforderungen, die sich immer wandeln und das auch noch auf letztlich unvorhersehbare Weise, ist das jedoch kontraproduktiv. Die Aufgabe des Product Owners ist nicht, früh und umfassend über Anforderungen Auskunft zu geben, sondern bewusst in kleinen, überraschenden Happen zu informieren. Die Softwareproduktion muss sich sozusagen selbst immer wieder verstören mit Unvorhergesehenem. Nur so wird sich die Programmierung hin zu Methoden und Strukturen gezogen fühlen, die dauerhaft wandelbar sind. Unvorhergesehenes darf nicht die Ausnahme, sondern muss die Norm sein.

Auch das ist keine leichte Aufgabe für den Product Owner. Und so wird unterstrichen, wie wichtig diese Rolle für die Softwareproduktion ist. Er stellt die zentrale Kraft für Entstehung und Erhalt von Nachhaltigkeit dar. Nachhaltigkeit des Gesamtsystems entwickelt sich, indem ein stetiger Sog für kleine, überraschende Anforderungen aufgebaut wird.

Ralf Westphal

WEB-TIPP:
www.ccd-school.de

Quelle:
[1] Ralf Westphal, Die wichtigste Rolle in der Softwareentwicklung, http://blog.ralfw.de/2014/06/die-wichtigste-rolle-in-der.html