LEAN SCRUM: DER NUTZEN VON LEAN THINKING FÜR DIE AGILE PRODUKTENTWICKLUNG

Während agile Vorgehensweisen in der Softwareentwicklung mittlerweile stark verbreitet sind, ist die Anwendung von Lean Thinking in der Produktentwicklung deutlich seltener vorzufinden. Bei Nokia Siemens Networks werden Lean und Agile in der Produktentwicklung zusammen genutzt, als Teil einer Strategie, flexibler und schneller auf geänderte Umstände reagieren zu können. In diesem Artikel beleuchten wir, warum das so ist. Wir zeigen auf, welche der von agilen Vorgehensweisen gelassenen Lücken mit Hilfe von Lean Thinking geschlossen werden.

die autoren

Sabine Canditt
(E-Mail: sabine.canditt@siemens.com)
unterstützt als Certified Scrum Trainerin, Expertin und Coach für Agilität Teams und Organisationseinheiten und engagiert sich für die agile Community innerhalb und außerhalb ihrer Firma.

Dr. Peter Braun
(E-Mail: peter.m.braun@nsn.com)
leitete bei Siemens Mobile Networks das erste Projekt, in dem ein agiler Entwicklungsprozess eingeführt wurde. Derzeit arbeitet er als Coach für Agilität und unterstützt die Einführung von agilen Methoden bei NokiaSiemensNetworks.

Als Beispiel betrachten wir ein agiles Softwareentwicklungsteam. Wir nehmen an, dass dieses Team eine Kombination aus Scrum und *XP*-Praktiken (*eXtreme-Programming*) anwendet. Unser Team kann dadurch einige Erfolge verzeichnen. Der *Product Owner (PO)* versteht es, seine Produktvision klar und deutlich unter die Leute zu bringen, sodass das Team stets den Wert seiner Arbeit vor Augen hat. Das Team definiert ein Sprint-Ziel zusammen mit dem PO auf Basis eines priorisierten *Product Backlogs (PB)*, das erkennbar von der Produktvision abgeleitet ist. Nachdem sich das Team auf ein Sprint-Ziel verpflichtet hat, kann es sich ungestört der Erzeugung eines potenziell auslieferbaren Produktinkrements widmen. Am Ende des Sprints bekommt das Team ein Feedback vom PO und manchmal sogar direkt vom Kunden. Jeder Sprint bietet Erfolgserlebnisse und auch die Möglichkeit zu Verbesserungen. Falls jemand versucht, das Team während des Sprints zu stören und es von der Erreichung seines Ziels abzuhalten (etwa der PO oder das Management), wehrt der ScrumMaster dies ab, um den Arbeitsfluss des Teams möglichst nicht zu unterbrechen und einen Produktivitätsverlust zu vermeiden. Der ScrumMaster hilft dem Team zusätzlich, indem er lästige Hindernisse aus dem Weg räumt.

Was bei agiler Entwicklung gut funktioniert

Unser Team organisiert sich selbst, d. h. es darf allein entscheiden, wie viel Arbeit es für einen Sprint annehmen kann. Es entscheidet auch über die Arbeitsaufteilung und die Vorgehensweise. Das Team arbeitet funktionsübergreifend (*cross functional*), d. h. alle zur Realisierung des Produktinkrements notwendigen Fähigkeiten sind im Team vorhanden. Die Teammitglieder arbeiten ausschließlich für dieses eine Team und sitzen gemeinsam in einem Raum. Es wird viel kommuniziert, sodass man mit einem Minimum an interner Dokumentation auskommt. Die Teammitglieder können sich dem widmen, was ihnen Spaß macht, nämlich qualitativ hochwertige Software zu entwickeln. Und das machen sie gut, denn sie wenden konsequent gute Engineering-Praktiken an. Automatische Tests auf allen Ebenen (vom Unit-Test bis zum Akzeptanztest) und ständiges Refaktorisieren garantieren eine hohe Codequalität, auf die das Team mit Recht stolz ist. Die strikte Einhaltung der *Done*-Kriterien vermeidet von Anfang an den Aufbau von „Qualitätsschulden", die später womöglich mit schwer änderbarem Code teuer bezahlt werden müssten. Wenn die Kompetenzen im Team nicht optimal verteilt sind, lernen die Mitglieder voneinander, z. B. durch das Arbeiten in Paaren. Überhaupt ist die Arbeit konstruktiv und bereichernd, da das Team gemeinsam die Verantwortung für das Sprint-Ziel übernommen hat. Wenn Probleme oder Konflikte auftreten, sorgt der ScrumMaster dafür, dass sie als Chance für Verbesserungen erkannt und genutzt werden.

Diese bewusst als „heile Welt" geschilderte Kombination einer konsequenten Anwendung von Scrum und XP bietet alles, um unser agiles Team erfolgreich und glücklich zu machen. Und auch der Kunde ist zufrieden mit den Ergebnissen, denn Agilität erlaubt es ihm, frühzeitig und regelmäßig Einblick in den Stand der Entwicklung zu nehmen und darauf Einfluss nehmen zu können – das ist besonders dann wichtig, wenn er zu Beginn noch gar nicht so genau weiß, was er will.

Welche Fragen offen bleiben

Ist mit dieser Vorgehensweise der langfristige Geschäftserfolg garantiert? Kann die Organisation damit in einem sich ständig ändernden Umfeld überleben?

Versetzen wir uns in die Lage des PO, der dafür verantwortlich ist, das *richtige* Produkt zum *richtigen* Zeitpunkt herauszubringen. Wie kommt es, dass die Entstehung eines Produkt-Release manchmal so lange dauert, obwohl unser Team so effizient und agil arbeitet? Wie kann der PO die Entwicklungsdauer oder die Kosten seines Produkts optimieren? Wie steht es mit der Rolle des ScrumMasters, die nicht mit Macht- und Entscheidungsbefugnissen ausgestattet ist? Seine Aufgabe ist es, die Produktivität des Teams zu erhöhen, indem er Hindernisse beseitigt. Wie geht das ohne Macht?

Diese Fragen führen uns zur Rolle des Managements. Welche Aufgabe haben die Manager in einer agilen Organisation?

Scrum schweigt sich darüber aus, die Rolle existiert in dem Framework gar nicht. Und was ist mit den anderen Stakeholdern, wie beispielsweise Qualitätssicherung, Service, Marketing, Logistik und Produktion? Der PO muss sie alle einbinden, wenn er den Produkterfolg gewährleisten will. Wann und wie der das macht, ist ihm überlassen.

Besonders deutlich wird die Problematik, wenn die Produkte große Hardware- und Mechanikanteile enthalten, wie es bei vielen Siemens-Produkten – zum Beispiel Zügen, Kraftwerken und Computer-Tomographen – der Fall ist. Der Software-Entwicklungsprozess ist nur Teil des Produktentstehungsprozesses, der zusätzliche Herausforderungen in sich birgt:

- Das Prinzip der kurzen Iterationen mit immer gleicher Länge (*Timeboxes*), an deren Ende ein potenzielles Produktinkrement steht, ist in der Regel auf die Hardware- und Mechanikteile nicht anwendbar.
- Änderungen im späteren Projektverlauf sind teuer oder sogar unmöglich. Nehmen wir beispielsweise die Herstellung eines Zuges: Sind die Pressen für die Blechteile eines Zuges erst einmal gefertigt, sind die Investitionen dafür nicht mehr rückgängig zu machen. Daher ist es notwendig, grundlegende Designentscheidungen rechtzeitig zu treffen. Man kann sich nicht darauf verlassen, dass die gesamte Architektur erst im Projektverlauf entsteht. Auch eine frühzeitige Planung zur Integration von Hardware, Software und Mechanik ist notwendig.
- Am Ende der Entwicklung ist eine Validierung erforderlich, die sich nicht auf die einzelnen Iterationen verteilen lässt. Diese Tests sind auch kaum zu automatisieren. Der finale Test des Zuges findet auf der Schiene statt, wenn er komplett montiert ist.
- Es ist besondere Sorge dafür zu tragen, dass das Produkt keinen Schaden an Menschen und Umwelt verursacht.
- Vor der Inbetriebnahme des Produkts sind oft spezielle Aktivitäten nötig (z. B. Zertifizierung, Übersetzung in verschiedene Sprachen, Anstoßen von Marketing- und Serviceaktivitäten in verschiedenen Ländern).
- Die Kosten für das Produkt werden durch Herstellungs- und Materialkosten dominiert, nicht durch den Aufwand für die Entwicklung. Auch das muss im Design berücksichtigt werden, wenn die Kosten ein Erfolgsfaktor sind.
- Der Lebenszyklus des Produkts erfordert es, dass man sich frühzeitig Gedanken über Logistik, Service, Updates, Ersatzteile, Zulieferer und Außer-Betriebnahme macht.

Bei dieser genauen Betrachtung entsteht der Eindruck, dass unser agiles Team sich ein lokales Optimum geschaffen hat: Die Softwareentwicklung wird immer weiter optimiert – aber welche Relevanz hat das für den Lebenszyklus des Produkts und für die gesamte Organisation? Scrum hält zur Lösung dieser Problematik zwei Ratschläge bereit:

1. *Funktionsbergreifende Teams:* Konsequent angewandt, könnte man alle Stakeholder in diese Teams mit einbinden, zumindest in den frühen Phasen der Produktentstehung. In der Praxis wird das unserer Erfahrung nach heute kaum gemacht. Ein reales Scrum-Team bindet im besten Fall die Expertise aus der Entwicklung – also Entwickler, Tester, Architekten, Dokumentationsexperten usw. – mit ein, aber nicht zum Beispiel die Hardwareentwicklung.
2. *Inspektion und Adaption:* Das ist ein starker Ansatz, der unserem Team dabei hilft, sein (lokales) Optimum zu erzeugen. Konsequent angewandt, kann man dasselbe Prinzip auch auf die gesamte Organisation übertragen. Wir sehen das auch teilweise in der Praxis, wo globale Retrospektiven dazu dienen, gute Praktiken über Teamgrenzen hinweg bekannt zu machen, und wo Probleme, die außerhalb des Wirkungsfeldes eines Teams liegen, konzentriert angegangen werden. Wir sehen aber auch, dass die Erfahrungen leicht versanden und die Probleme ungelöst bleiben.

Was kann Lean zum agilen Erfolgsteam beitragen?

Die Begriffe *Lean* oder *Lean Thinking* (vgl. [Wom03]) bezeichnen das Denkgebäude und die Prinzipien, die die Firma Toyota in den letzten 60 Jahren entwickelt hat und auf alle Unternehmensprozesse anwendet. Dadurch hat Toyota Maßstäbe bei Produktivität, Qualität und Profitabilität gesetzt[1]). Ziel von Lean ist es, immer mehr Wert für Kunden bei gleichzeitig immer geringeren Kosten zu schaffen. Die Grundideen von Lean sind *Respekt vor Menschen* und *Kontinuierliche Verbesserung*. Das wird auch als das *Lean-Gebäude* bezeichnet (siehe Abbildung 1).

Die gemeinsame Betrachtung von Lean und Agile bietet folgende Vorteile:

- Lean hat die gesamte Wertschöpfungskette im Blick und hilft bei der Optimierung des gesamten Produktlebenszyklus – von

[1]) Die kürzlich bekannt gewordenen Qualitätsprobleme werden bei Toyota werden von Experten nicht als ein Scheitern des Lean-Ansatzes gewertet, sondern darauf zurückgeführt, dass Toyota in den vergangenen Jahren zu schnell gewachsen ist und seine eigenen Prinzipien dadurch vernachlässigt hat (vgl. [Lik10]).

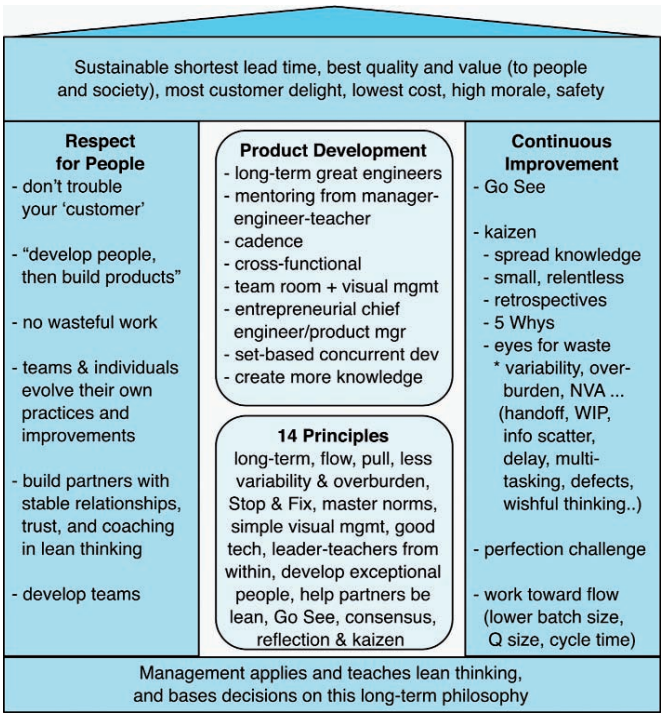

Abb. 1: *Das Lean-Gebäude mit den Grundideen „Respekt vor Menschen" und „Kontinuierliche Verbesserung" (aus [Lar09]).*

der Idee über die Produktion bis zum Service. Dadurch wird einer lokalen Optimierung des Softwareentwicklungsanteils entgegengewirkt.
- Lean hilft zu verstehen, warum agile Konzepte funktionieren und welche Voraussetzungen dafür gegeben sein müssen. Viele der Lean-Prinzipien (z. B. Begrenzen der Arbeitsmenge auf die Kapazität) oder Werkzeuge (z. B. Wertstrom-Analyse) helfen bei der konkreten Anwendung oder Optimierung eines agilen Prozesses. Die Vorgehensweisen von Lean und Agile haben sich unabhängig voneinander in ähnlicher Weise entwickelt und ergänzen sich gut. Beim Lean-Software-Development werden die Lean-Prinzipien direkt auf die Softwareentwicklung übertragen (vgl. [Pop06]).

Lean ist keine einfache Sammlung von Tools, die man einzeln einführen kann – vielmehr erfordern die Prinzipien oft grundlegende Änderungen in der Unternehmenskultur und -führung. Die meisten der Prinzipien ergänzen und verstärken sich gegenseitig, weshalb ein ganzheitlicher Ansatz nötig ist. Viele Versuche, Lean in Unternehmen einzuführen, sind nicht erfolgreich, weil nur Bausteine ausgewählt werden, die grundlegenden Managementprinzipien aber unverändert bleiben. Auch hier gibt es also Ähnlichkeiten von Lean und Agile.

Im weiteren Verlauf des Artikels bringen wir konkrete Beispiele für eine erfolgreiche Symbiose von Lean und Agile. Der Hauptfokus liegt dabei auf der Anwendung von Lean auf die Produktentwicklung. Lean wird auch in anderen Unternehmensprozessen (z. B. Produktion, Service, Vertrieb) und in vielen Unternehmen aus unterschiedlichsten Branchen angewendet (z. B. Gesundheitswesen, Einzelhandel, Luft- und Raumfahrt). Während agile Ansätze im Wesentlichen auf die Produktentwicklung ausgerichtet ist, betrachtet Lean das gesamte Unternehmen (**siehe Abbildung 2**).

Für die offenen Punkte, die wir im ersten Teil dieses Artikels identifiziert haben, sind folgende Lean-Ansätze interessant:

- Führungsverhalten und Entwicklung der Mitarbeiter
- Kontinuierliche Verbesserung auf Basis standardisierter Prozesse
- Qualitätsbewusstsein

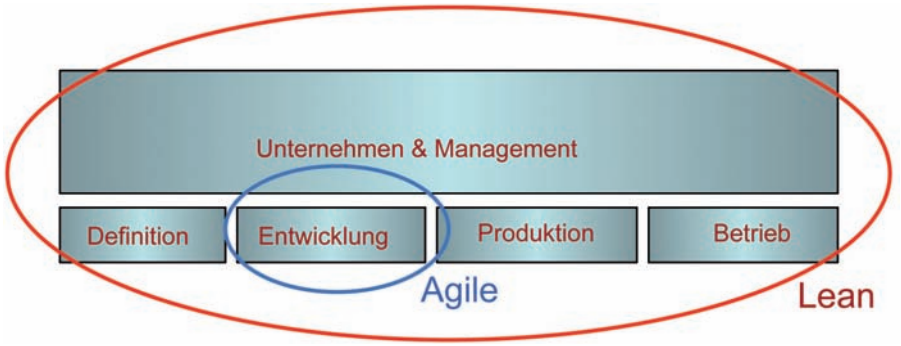

Abb. 2: *Unterschiedlicher Fokus von Lean und Agile.*

- Wertorientierung und Vermeidung von Verschwendung[2])
- Optimierung der Durchlaufzeit und des Durchsatzes durch Anwendung von *Pull* anstatt von *Push*
- Umgang mit Zulieferern
- Parallele Entwicklung von Alternativen (*Set-based Design*)
- Zusammenarbeit mit anderen Abteilungen in der Organisation

Einige der Lean-Prinzipien werden von unserem agilen Team bereits umgesetzt:

- Es ist die Aufgabe des PO, bei der Priorisierung des PB den Wert zu berücksichtigen und Verschwendung dadurch zu vermeiden, dass nur Anforderungen mit erkennbarem Wert analysiert und umgesetzt werden. Dadurch wird einer der folgenschwersten Arten von Verschwendung entgegengewirkt: der Realisierung unnötiger Anforderungen.
- Die Qualitätsanforderungen erreicht unser Scrum-Team durch die konsequente Einhaltung der *Done*-Kriterien am Ende jeden Sprints sowie durch testgetriebene Entwicklung und automatische Tests, die in jedem Sprint wiederholt werden. Dadurch werden Fehler frühzeitig gefunden – ein weiterer wichtiger Schritt, um Verschwendung zu vermeiden.
- Durch die funktionsübergreifende Aufstellung des Teams und die kurzen Zyklen können viele Übergaben, Dokumente und Verzögerungszeiten für halbfertige Arbeiten (*Work in Process*) vermieden werden.
- Das Team „zieht" sich die Arbeit für den nächsten Sprint aus dem PB und entscheidet selbst über die Menge der Arbeiten, statt dass ihm mehr Arbeit, als möglich ist, „aufgedrückt" wird. Das Pull-Prinzip führt dazu, dass die Menge der Arbeiten die Kapazität nicht übersteigt. Dadurch entsteht ein gleichmäßiger Fluss von Aktivitäten, der lange Warteschlangen zwischen einzelnen Aktivitäten und damit lange Verzögerungen vermeidet und den Durchsatz optimiert. Das Vermeiden von Überlast führt auch meist zu einer geringeren Fehlerrate und Qualitätsproblemen.
- Das Team verbessert sich kontinuierlich durch Retrospektiven am Ende jedes Sprints.

Es sieht also so aus, als ob Lean und Agil sich recht gut vertragen würden. Schauen wir uns nun an, welche Ergänzungen die Lean-Prinzipien bereithalten und welche Vorteile sich daraus für die Softwareentwicklung, aber auch die Gesamtproduktentwicklung ergeben.

Führungsverhalten

Lean bzw. agil zu werden, ist mit einem grundlegenden Wandel von Werten, Einstellungen und Verhalten verbunden. Dieser Änderungsprozess erfordert die Beteiligung des Managements. Bei Lean haben die Manager gleichzeitig die Rolle von Mentoren („Manager-Teacher"), die selbst Experten im Fachgebiet sind und ihre Mitarbeiter coachen. Sie verschaffen sich an Ort und Stelle einen Überblick über die Arbeit und die Probleme der Mitarbeiter (*Go see*), helfen mit, die Probleme zu lösen, und begründen so ein starkes Vertrauensverhältnis. Jedes Problem dient auch dazu, die Mitarbeiter zu Problem-

[2]) *Verschwendung ist im Sinne von Lean alles, wofür der Kunde nicht bereit ist zu zahlen.*

lösern auszubilden, die ständig Verbesserungen suchen und sich weiterentwickeln.

Diese Beschreibung erinnert an die Aufgaben des ScrumMasters und ergänzt diese sehr gut, da sie mit der nötigen Durchsetzungskraft ausgestattet ist. Der ScrumMaster kann wesentlich erfolgreicher arbeiten, wenn er Rückhalt bei einem Lean Manager findet. Der ScrumMaster hat keine formale Autorität und soll trotzdem Probleme beseitigen. Wie soll das gehen? Beim A3-Prozess[3]) von Lean (vgl. [Sho08]) wird Mitarbeitern die Verantwortung für Probleme übertragen, für die sie keine Entscheidungshoheit haben. Die Lösung erfordert meist das Einbeziehen anderer Organisationen. Nur wirklich sinnvolle Lösungen, die auch die anderen Beteiligten überzeugen, haben eine Chance auf Realisierung. Daher wird den Mitarbeitern ein Manager als Mentor zur Seite gestellt, der die notwendigen Türen öffnet.

Qualitätsbewusstsein

Fehler sind eine der größten Quellen von Verschwendung, da sie Nacharbeit erfordern. Daher sollten sie möglichst vermieden bzw. so früh wie möglich erkannt werden. Auch wenn z. B. Systemtests am Ende einer Entwicklung notwendig sind, ist es nicht das Ziel, dort Fehler zu finden, sondern dort möglichst *keine* Fehler mehr zu finden. Ein *Done*-Kriterium, das automatisierte Tests für alle neuen und alten Features enthält, verhindert, dass Fehler die Iteration unentdeckt verlassen. Besser noch ist eine testgetriebene Entwicklung für Akzeptanz- und Unit-Tests und auch *Pair Programming*, da hier Fehler bereits bei der Entstehung erkannt werden.

Diesem Ziel dient auch die frühzeitige Integration von Software mit elektronischen und mechanischen Komponenten, die gleichzeitig entwickelt werden. Werden die Schnittstellen sukzessive festgelegt, lassen sich z. B. Software- und Prototypen der Hardware frühzeitig integrieren und so über die Simulation der Hardware hinausgehende Erkenntnisse gewinnen.

Lean Thinking hält nichts von provisorischen Reparaturmaßnahmen. Bei der Lösung von Problemen geht man von der Annahme aus, dass die Ursache eines Fehlers meist im System oder im Prozess liegt und

[3]) *So benannt, weil die Problembeschreibung so einfach sein soll, dass sie auf ein Din-A3-Blatt passt.*

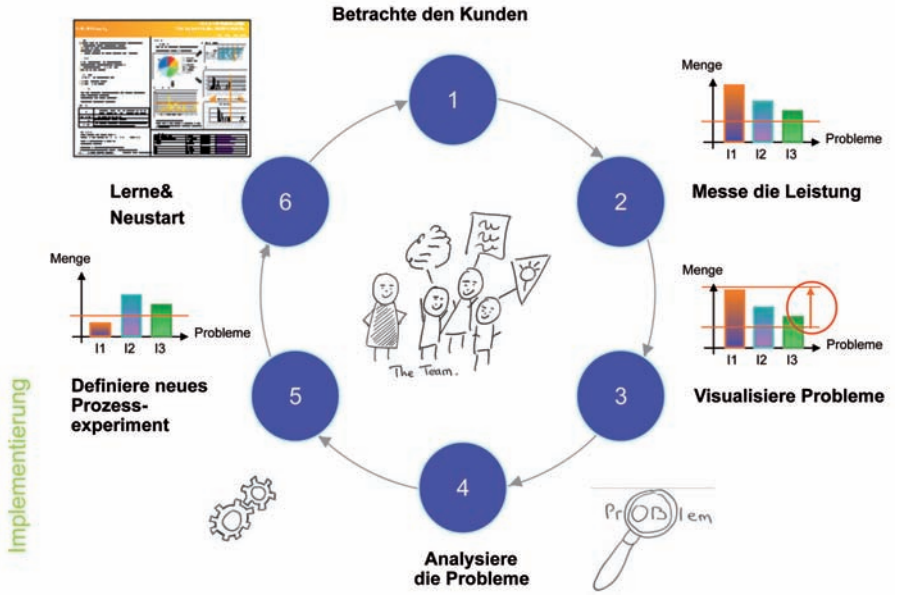

Abb. 3: PDCA im Einsatz.

dass daher das Auftreten eines Fehlers genutzt werden sollte, um die Schwachstelle im Prozess zu erkennen und zu beseitigen. Bei Scrum wird am Ende jeder Iteration eine Retrospektive durchgeführt, bei der die Ursache für Probleme ermittelt werden kann, beispielsweise durch wiederholtes Fragen von *Warum* (*5 Why*, eine weitere Lean-Praktik). Probleme im eigenen Umfeld kann das Team selbst beseitigen. Probleme außerhalb sollten zusammen mit dem ScrumMaster angegangen werden, was wiederum die Unterstützung des Managements erfordert.

Lean Thinking ist auf Langfristigkeit und Nachhaltigkeit ausgerichtet. Ein erfolgreiches Produkt zeichnet nicht nur den Profit aus, den ein Unternehmen damit macht, sondern auch dadurch, ob der Kunde das Produkt wieder kaufen würde und es weiterempfiehlt. Wenn der Kunde z. B. sein Produkt lange nicht benutzen kann, weil ein Ersatzteil nicht geliefert wird, trägt das nicht zu seiner Zufriedenheit bei. Ebenso wenig wird es ihn erfreuen, wenn er bei dem Versuch, sich Hilfe bei einer Service-Hotline zu holen, in endlosen Warteschleifen landet. Zusätzlich ist zu überlegen, wie das Produkt außer Betrieb genommen wird, welche Teile recycelt werden können usw. Die Rolle des PO ist in Scrum für den Geschäftserfolg des Produkts verantwortlich, in der Praxis endet die Verantwortlichkeit jedoch häufig mit der Abnahme durch den Kunden. Das kann jedoch nicht im Sinne der Organisation sein. Daher bezieht ein Lean PO die Anforderungen aus späten Phasen im Lebenszyklus unbedingt mit ein.

Kontinuierliche Verbesserung auf Basis standardisierter Prozesse

Lean Thinking geht davon aus, dass diejenigen Menschen, die die Arbeit selbst verrichten, diese am besten beurteilen und verbessern können. In der Praxis bewährte Vorgehensweisen werden standardisiert, um den erreichten Wissensstand einzufangen. Diese Standards stellen aber nur den augenblicklich besten Kenntnisstand dar und dienen als Basis für weitere Verbesserungen. Ziel ist es, durch ständige Verbesserung dem Ziel „Perfektion" immer näher zu kommen. Standards müssen zwar befolgt werden, aber auch genügend Freiraum zum Experimentieren lassen.

Jeder ist aufgefordert, Probleme zu lösen und zur Prozessverbesserung beizutragen. Die Prozesse entstehen daher immer von unten nach oben, also aus der erprobten Praxis heraus. Lean-Prozessverbesserung erfolgt in einem Zyklus mit den folgenden vier Phasen, die durch W. E. Deming (vgl. [Dem89]) bekannt wurden:

- *Plan:* Erkennen von Verbesserungspotenzialen, die Analyse des aktuellen Zustands, Entwickeln eines neuen Konzepts
- *Do:* Ausprobieren und praktische Optimieren des Konzepts mit einfachen Mitteln im Kleinen

- *Check:* Überprüfung der Ergebnisse und – bei Erfolg – Umsetzung auf breiter Front als Standard
- *Act:* Einführung auf breiter Front, Überprüfung auf Einhaltung

Abbildung 3 zeigt ein Beispiel für die Anwendung eines *PDCA*-ähnlichen (*Plan Do Check Act*) Verbesserungszyklus bei einem Team von Nokia Siemens Networks. In jedem Zyklus werden ein oder mehrere potenzielle Verbesserungen eingeführt. Erweist sich der Ansatz als positiv, was anhand von Metriken überprüft wird, wird das Vorgehen Teil des Standardprozesses.

Die Lösung von Problemen obliegt den Personen, die das Problem haben, weil diese auch das größte Interesse und meist auch das beste Problemverständnis daran haben. Dabei sind sie allerdings nicht auf sich allein gestellt, sondern ein Manager agiert als Mentor. Diese ständige Verbesserung wird auch als Toyotas *Kata* bezeichnet (vgl. [Rot09]).

Die Beurteilung der Wirksamkeit von Prozessverbesserungen schließt auch mit ein, globale Daten zu sammeln und zu analysieren, um den Effekt von Prozess-Experimenten beurteilen zu können. Sinnvolle Metriken, die das „große Ganze" widerspiegeln, sind die Durchlaufzeit, Kundenzufriedenheit oder die Anzahl der Fehler.

Die Scrum-Retrospektive kann als eine Form des PDCA-Modells verstanden werden. In der Retrospektive werden Prozessexperimente entwickelt. Stellen die Veränderungen eine Verbesserung dar, werden sie beibehalten, andernfalls werden sie verworfen. Die Standardisierung und die Einführung einer kontinuierlichen Verbesserung auf breiter Front gehen über die Retrospektiven unseres agilen Teams deutlich hinaus. In der Praxis werden in großen Organisationen agile Retrospektiven skaliert, um gute Praktiken auch anderen Teams bekannt zu machen und um Probleme anzugehen, die ein einzelnes Team nicht lösen kann. Scrum äußert sich jedoch nicht dazu, wie das Wissen tatsächlich festgehalten wird. Von Lean können wir lernen, globale Daten für die Prozessverbesserung heranzuziehen, statt uns ausschließlich auf unser Bauchgefühl zu verlassen.

Ein weiterer Unterschied zwischen Lean und Agile ist, dass die Prozessverbesserung im Lean Thinking keine Grenzen kennt. Wenn die Praktiken „aus dem EffEff" beherrscht werden, löst sich der erfahrene Praktiker von ihnen und geht über sie hinaus. Dagegen wird in Scrum davon abgeraten, das Scrum-Framework selbst zu ändern oder zu verlassen.

Verbesserungen, die die eigene Arbeit in Frage stellen oder überflüssig machen, werden nur dann vorgeschlagen, wenn die Mitarbeiter darauf vertrauen können, dass sie sich damit nicht um Kopf und Kragen bringen, sondern dass gegebenenfalls neue Aufgaben im Unternehmen auf sie warten. Die Investition in die Mitarbeiterschaft muss in der Unternehmenskultur ein hohes Gut darstellen – wie es ja in den Lean-Grundwerten verankert ist (Respekt vor Menschen, kontinuierliche Verbesserung).

Wertorientierung und Vermeidung von Verschwendung

Wert ist im Sinne von Lean alles, wofür der Kunde zu zahlen bereit ist. Alles andere ist Verschwendung. Manche Arten von Verschwendung sind (noch) notwendig, wenn wir sie derzeit benötigen, um wertschöpfend arbeiten können. Das Ziel ist es, diese Verschwendung so weit wie möglich zu beseitigen. Lean stellt daher das Ziel, den Kundennutzen zu maximieren, immer in den Vordergrund.

Wir haben gesehen, dass unser Scrum-Team bereits auf einem guten Weg ist, Verschwendung zu vermeiden. Lean Thinking kann jedoch noch weiter helfen, den Blick für Verschwendung zu schärfen. Mary Poppendieck (vgl. [Pop06]) hat sieben wichtige Arten der Verschwendung von der Produktion (vgl. [Shi81]) auf die Softwareentwicklung übertragen:

- *Work in Process:* Angefangene und nicht fertig gestellte Arbeiten, z. B. analysierte, aber noch nicht realisierte Anforderungen. Diese halbfertigen Arbeiten werden in Warteschlangen zwischengelagert und blockieren dort den Durchfluss für neue Arbeiten. Dies führt zu langen Verzögerungen. Unser Scrum-Team achtet beispielsweise darauf, dass es nicht mit allen PB-Items erst einmal anfängt und diese dann parallel bearbeitet, sondern möglichst nur wenige gleichzeitig in Arbeit hat.
- *Überproduktion:* Features, die keiner benötigt. Das stellt nicht nur eine Verschwendung von Aufwand bei der Entwicklung dar, sondern langfristig entsteht dadurch eine enorme Komplexität, die dazu führt, dass sich die Software immer schwerer warten, ändern und weiterentwickeln lässt. Diese Verschwendung vermeidet Scrum durch die Fokussierung auf den priorisierten PB. Der PO muss dafür sorgen, dass der Backlog nicht ungehemmt anwächst.
- *Verloren gegangenes Wissen:* Das agile Team verlässt sich zu großen Teilen auf das Wissen in den Köpfen. Das funktioniert, da die verschiedenen Aktivitäten von der Analyse über die Realisierung bis zum Test zeitnah in einem Sprint stattfinden. Sonst bestünde die Gefahr, dass tätsächlich Wissen, z. B. über die wichtigsten Randbedingungen für Anforderungen, verloren geht.
- *Übergaben, z. B. von der Entwicklung zum Test:* In Scrum arbeiten Entwickler und Tester eng zusammen in einem Team. Sie arbeiten an einem gemeinsamen Ziel und kommunizieren täglich miteinander. Dieser direkte Kontakt vermeidet Übergaben von einer Abteilung in die andere, die zusätzliche Dokumentation und gegenseitige Absicherung erfordern würde.
- *Multitasking:* Beispielsweise dadurch, dass Teammitglieder in mehreren Projekten gleichzeitig arbeiten und dadurch Aufgaben erst verzögert fertigstellen können, was die Leistungsfähigkeit stark verringert. Um dies zu vermeiden, sollten die Teammitglieder möglichst mit ihrer gesamten Arbeitszeit in einem Team mitarbeiten.
- *Verzögerungen:* Beispielsweise warten auf Entscheidungen und Informationen. Hier helfen wieder eine zeitnahe Erledigung und ein funktionsübergreifendes Arbeiten im Scrum-Team.
- *Fehler, die Korrekturen und Nacharbeiten erfordern:* Diese werden vermieden durch die bereits oben beschriebenen Testansätze und die konsequente Einhaltung der *Done*-Kriterien.

Wenn wir überflüssige Verschwendung aufspüren und beseitigen wollen, müssen wir uns nicht nur auf das Team selbst, sondern vor allem auf die Schnittstellen konzentrieren.

Bei der Zusammenarbeit mit Hardware- und Mechanik-Anteilen des Produkts kann man durch kontinuierliche Zusammenarbeit (*Simultaneous Engineering*) z. B. Verschwendung von Überproduktion vermeiden, weil die genauen Anforderungen gemeinsam erarbeitet werden und möglichst am konkreten Objekt überprüft werden. Auch Experten aus der Produktion und Zulieferer sollten frühzeitig in den Designprozess einbezogen werden, um spä-

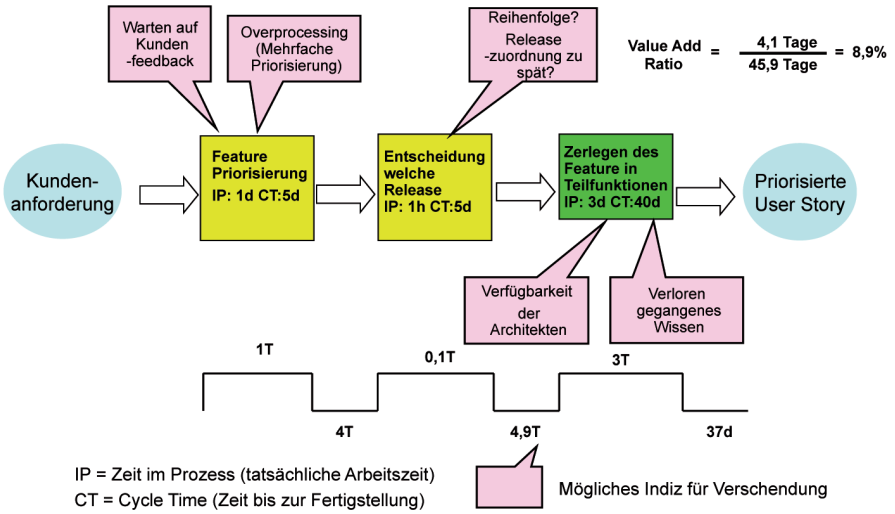

Abb. 4: Wertstrom-Analyse für die Voranalyse-Phase einer Kundenanforderung.

ter Kosten bei der Produktion sparen bzw. die Abläufe beschleunigen zu können. Wichtige Praktiken sind Wiederverwertung und Verwendung von Standardteilen. Diese Experten helfen auch mitzubestimmen, welche Entscheidungen wann getroffen werden müssen, um einerseits die wesentlichen architektonischen Entscheidungen rechtzeitig zur Verfügung zu haben und andererseits möglichst viele Freiheiten für Optimierung zu behalten.

Die Wertstrom-Analyse dient der Identifikation von Verschwendung in einem Prozess. Dazu betrachtet man die gesamte Wertschöpfungskette, z. B. von der Idee bis zum fertigen Produkt oder vom Fehlereingang bis zur ausgelieferten Korrektur. Am Anfang und am Ende steht jeweils der Kunde. Bei dieser Analyse werden alle Zeiten erfasst, in denen aktiv gearbeitet wird, und auch die Wartezeiten. Das Ergebnis zeigt häufig, dass die Verzögerungszeiten außerhalb des Teams an den Schnittstellen zu anderen Disziplinen liegen.

Die Wertstrom-Analyse zeigt den tatsächlich gelebten Prozess und nicht den idealisierten Ablauf aus dem Prozesshandbuch. Aus den identifizierten Verbesserungsmöglichkeiten wird der zukünftige Wertstrom aufgezeigt und es werden Maßnahmen definiert, um die Verbesserungen umzusetzen.

Abbildung 4 zeigt als Beispiel die Wertstrom-Analyse für die Priorisierung von Kundenanforderung und Aufteilung auf Software-Releases. Die Wünsche verschiedener Kunden werden priorisiert, bevor diese an die Entwicklung weitergegeben werden, werden. Die Bereiche, bei denen eine potenzielle Verschwendung identifiziert wird (zum Beispiel Verzögerung dadurch, dass die Architekten nicht rechtzeitig eingebunden werden), werden analysiert, um so die Durchlaufzeit reduzieren zu können.

Optimierung der Durchlaufzeit

Jeder kennt den Effekt: Plötzlich kommt es auf der Autobahn zu einem Stau, obwohl der Verkehr doch eben noch so schön im Fluss war. Die Ursache dafür ist eine Überlastung. Genauso ist es auch mit unserem Entwicklungsteam: Wenn es überlastet ist, müssen die Kunden lange auf die Umsetzung ihre Anforderungen warten. In der traditionellen Entwicklung hat man häufig das Ziel, die Auslastung der Mitarbeiter zu maximieren. Die Warteschlangen-Theorie zeigt, dass genau das zu einer Erhöhung der Durchlaufzeiten führt.

Die Kurve in Abbildung 5 (vgl. [Pop06]), die sich aus der Warteschlangen-Theorie ermitteln lässt, veranschaulicht dies. Die Durchlaufzeit wächst relativ moderat bis zu einer Auslastung von 80 % und danach sehr steil. Dieser Effekt ist um so stärker, je größer die Losgrößen der zur Verarbeitung anstehenden Aufgaben sind, und je größer die Variation beim Eintreffen dieser Aufgaben ist. Kurze Iterationen und gleichermaßen kleine Arbeitsaufgaben, wie wir sie im Sprint-Planning bearbeiten, helfen also auch, die Durchlaufzeiten zu verringern.

Zwar kann das Team selbst entscheiden, wie viel es in einem Sprint umsetzen kann, aber dadurch vermeidet es nicht einen Rückstau im PB. Vielleicht ist der Kunde beruhigt, dass seine Anforderungen überhaupt erfasst wurden, aber eine Analyse, wie lange es dauern würde, den kompletten Backlog zu realisieren, bringt oft Ernüchterung. Viel sinnvoller ist es, nur so viel Arbeit für ein Team oder eine gesamte Entwicklungsabteilung im Vorrat zu halten, wie auch in absehbarer Zeit erledigt werden kann.

Ein ähnlicher Effekt tritt auch bei der Laufzeit von Projekten ein: Organisationen, die viele parallele Projekte haben, erzeugen Verschwendung, wenn Mitarbeiter in mehreren Projekten gleichzeitig arbeiten müssen. Außerdem stauen sich Projekte vor Flaschenhälsen (z. B. Test-

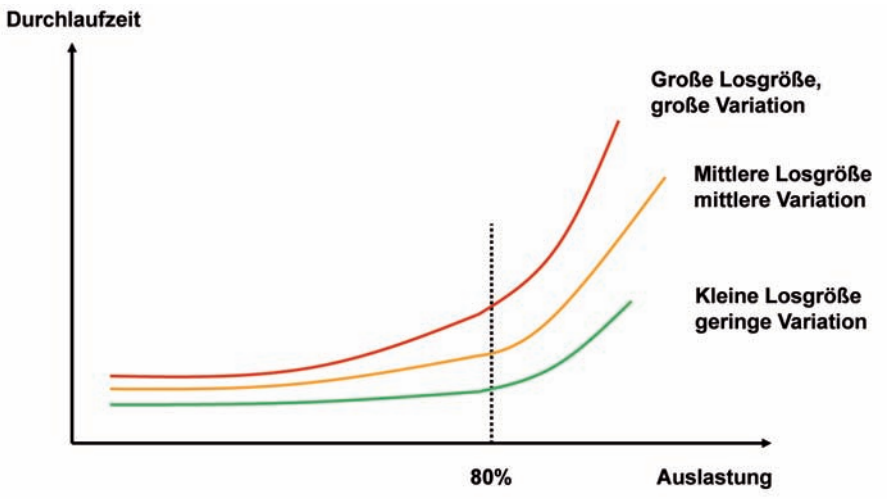

Abb. 5: Durchlaufzeit als Funktion der Auslastung.

maschinen) und verlängern so die Laufzeiten für alle Projekte.

Bei Lean wird deshalb nicht die Auslastung optimiert, sondern die Durchlaufzeit. Die Kapazität in den einzelnen Prozessschritten wird so angepasst, dass ein möglichst gleichmäßiger Fluss entsteht. Es wird nur so viel erzeugt, wie die nächste Station auch abnehmen kann. Die Steuerung bei Produktionsprozessen erfolgt häufig durch so genannte Kanban-Karten, die vom Abnehmer geschickt werden, wenn neue Ware produziert werden soll.

Die Kanban-Karten gaben auch dem Kanban-Verfahren den Namen, das von agilen Teams vor allem für die Software-Wartung eingesetzt wird. Bei Kanban werden die aktiven Aufgaben der verschiedenen Stufen im Prozess (z. B. Analyse, Entwicklung, Freigabetest) auf einem Whiteboard mit Karten visualisiert. Die Zahl der gleichzeitig im System befindlichen Aufgaben wird je Stufe begrenzt, um so den Durchsatz zu optimieren (vgl. [And10], [Kni10]).

Umgang mit Zulieferern

Der Respekt gegenüber den Menschen, ein Grundpfeiler des Lean Thinking, schließt natürlich auch Zulieferer mit ein. Ziel ist es, eine langfristige Beziehung zu Zulieferern zu schaffen, die es erlaubt, gemeinsam miteinander zu lernen. Dazu ist es sinnvoll, Zulieferer frühzeitig in den Designprozess und auch in Retrospektiven mit einzubeziehen.

Set-based Design

Unter *Set-based Design* versteht man die parallele Entwicklung von Lösungsalternativen. Bedeutet das nun, dass mit doppeltem Aufwand gearbeitet wird? Im ersten Moment mag es wie Verschwendung aussehen. Auf den zweiten Blick zeigt sich aber der deutliche Vorteil: Set-based Design erlaubt es, zu einem bestimmten Zeitpunkt die beste von mehreren untersuchten Designalternativen verfügbar zu haben. Die Alternative wäre, alles auf eine Karte zu setzen – aber was ist, wenn diese Karte nicht funktioniert? Dann steht man ohne Lösung da und muss unter Umständen von vorne anfangen. Die dadurch entstehende Verzögerung kann sehr teuer werden, z. B. wenn ein Konkurrent schneller auf den Markt kommt – viel teurer als der vermehrte Aufwand durch die Parallelentwicklung. Set-based Design ist eine Möglichkeit für den PO, die Entwicklung zu beschleunigen. Diese Methode ist besonders geeignet für nicht-iterative Entwicklung, sie ist aber auch für Softwareteile sinnvoll (z. B. bei der Entwicklung von Alternativen zu einer Benutzungsschnittstelle, aus denen der Kunde sich eine aussuchen kann, oder bei der Evaluierung von Architekturen im Hinblick auf nicht-funktionale Anforderungen).

Zusammenarbeit mit anderen Entwicklungsabteilungen

Um die Entwicklungszeit über alle Bereiche zu verkürzen, wird beim Lean Development ein so genannter *Obeya* (japanisch für „großer Raum") genutzt, in dem die Projektleiter der verschiedenen Teilbereiche einen Arbeitsplatz haben. Im Obeya sind alle für das Vorhaben relevanten Planungs- und Entwurfsinformationen ausgestellt (visuelles Management). Die Zusammenarbeit über Abteilungsgrenzen hinweg ermöglicht es, sehr schnell Entscheidungen zu treffen oder Probleme zu adressieren, die alle an der Entwicklung des Produkts beteiligten Bereiche betreffen. Dies ist die nächste Stufe von agilen Teams, die in einem Raum sind, um so eine möglichst gute Zusammenarbeit zu ermöglichen (vgl. [Mor06]).

Zusammenfassung

Lean bietet einen erweiterten Blickwinkel über die Grenzen einzelner Teams auf die gesamte Organisation und die gesamte Wertschöpfungskette. Die agile Vorgehensweise kann als Möglichkeit angesehen werden, um Lean-Prinzipien in der Softwareentwicklung umzusetzen. Schneller als die Wettbewerber zu lernen, ist der wichtigste strategische Vorteil von Lean-Unternehmen. Die Lean-Prinzipien können dabei helfen, Schwachstellen in der eigenen Vorgehensweise zu erkennen. Der wichtigste Gewinn von Lean und Agile besteht darin, eine Unternehmenskultur zu schaffen, in der Mitarbeiter und Führungskräfte lernen, sich ständig weiter zu verbessern und damit für die Herausforderungen der Zukunft gut gerüstet zu sein. ■

Literatur & Links

[And10] D. Anderson, Kanban, Blue Hole Press 2010

[Dem89] W.E. Deming, Out of the Crisis, MIT 1989

[Kni10] H. Kniberg, M. Skarin, Kanban and Scrum making the best of both, 2010, siehe: http://www.infoq.com/minibooks/kanban-scrum-minibook

[Lar09] C. Larman, B. Vodde, Lean Primer, 2009, siehe: http://www.leanprimer.com/downloads/lean_primer.pdf

[Lik10] J. Liker, The Wrong Lessons From Toyota's Recalls – And the Truth, 2010, siehe: http://blogs.hbr.org/cs/2010/03/dont_believe_everything_you_re.html

[Mor06] J. Morgan, J. Liker, The Toyota Product Development System : Integrating People, Process, and Technology, Productivity Press 2006

[Pop06] M. und T. Poppendieck, Implementing Lean Software Development: From Concept to Cash, Addison Wesley Professional, 2006

[Rot09] M. Rother, Toyota Kata: Managing People for Improvement, Adaptiveness and Superior Results, McGraw-Hill, 2009

[Shi81] S. Shingo, Study of „Toyoda" Production System from an Industrial Engineering Viewpoint, Productivity Press, 1981, Chapter 5

[Sho08] J. Shook, Managing to Learn, Lean Enterprise Institut, Using the A3 management process 2008

[Wom03] J. Womack, D. Jones, Lean Thinking: Banish Waste and Create Wealth in Your Corporation, Free Press 2003 (2.Aufl.)