2 Von rechts nach links in der digitalen Welt

Wir befinden uns in den Büros von Springboard DIY, einem Onlinehändler von Heimwerkerprodukten. Alex, eine Senior-Entwicklerin, trifft gerade den Produktmanager Rowan auf dem Flur. Rowan ist auf dem Weg zu einer externen Besprechung.

Alex: Ich bin so froh, dass ich dich noch erwischt habe, bevor du losgehst. Wir sind gerade dabei, die Änderungen für die Elektrowerkzeuge dauerhaft zu implementieren, aber es gibt einige Dinge, die ich zuerst mit dir besprechen möchte. Können wir uns demnächst mal kurz zusammensetzen?

Rowan: Natürlich! Gibt es etwas, worüber ich mir Gedanken machen sollte? Alles, was ich seit dem letzten Release in den Daten gesehen habe, sieht bisher großartig aus – mehr Menschen navigieren schneller durch diese Seiten und sie kaufen ein!

Alex: Keine Sorge, das sehe ich auch so. Ich frage mich jedoch, ob es hier nicht noch eine Möglichkeit gibt, die wir gerade verpassen. Es wäre sicher sinnvoll, sich mal damit zu befassen, solange wir es noch frisch im Gedächtnis haben.

Rowan: Klingt verlockend! Ich muss jetzt aber los, wie sieht es bei dir um 14 Uhr heute Nachmittag aus? Bis dahin bin ich wieder zurück im Büro.

2.1 Was ist gerade passiert?

Wir haben gerade ein kleines Beispiel dafür gesehen, was passieren kann, wenn Menschen bei der schnellen Entwicklung von funktionsfähiger Software eng zusammenarbeiten, und zwar so, dass die Software bereits Mehrwert für den Kunden bringt. Das ist es, was wir sehr schnell in jeder Organisation erwarten sollten, die diesen Dingen einen hohen Stellenwert einräumt:

- **Kollaboration –** sowohl intern (innerhalb und zwischen Teams) als auch extern (vor allem mit Kunden)
- **Kontinuierliche Auslieferung –** funktionsfähige Software (oder funktionsfähige Produkte und Dienstleistungen), die bereits irgendeine Art von Bedarf erfüllt und sich durch schnelle Abfolgen kleiner *inkrementeller* (additiver) Erweiterungen und *iterativer* (wiederholter) Verfeinerung weiter verbessert [29], sowie die Infrastruktur, um diese Änderungsrate effizient zu erreichen [30].
- **Anpassungsfähigkeit –** nicht nur Flexibilität in Bezug auf Prioritäten und Zeitpläne, sondern auch die Möglichkeit, die Gestaltung von Produkten, Prozessen und der Organisation als Reaktion auf (oder in Erwartung von) sich ändernde Umstände und neues Wissen anzupassen. Nicht zuletzt auch durch die im Lieferprozess gewonnenen Erkenntnisse.

In einer Entwicklungsumgebung laufen drei Prozesse gleichzeitig ab:

1. Die relativ vorhersehbare Aufgabe, gut verstandene Anpassungen durch einen vorwiegend linearen Prozess zu liefern, wobei die Ergebnisse in einzelnen Durchgängen durch das System schnell erreicht werden.
2. Die Entwicklung des Produkts durch einen emergenten und erkenntnisfördernden Prozess, der das Potenzial hat, alles, was ihn betrifft, auf unvorhersehbare Weise zu beeinflussen; die Ergebnisse werden mit Geduld durch diszipliniertes Experimentieren erzielt.
3. Entwicklung einer Infrastruktur aus Technologie, Verfahren und Kultur, die zur Unterstützung der ersten beiden Prozesse benötigt wird.

Ein wesentlicher Unterschied zur typischen Fabrikumgebung der materiellen Welt besteht darin, dass an allen drei Prozessen die gleichen Personen beteiligt sind. In einer gesunden Produktentwicklungsumgebung werden diese drei Prozesse im Gleichgewicht gehalten. Wenn sie aus dem Gleichgewicht geraten, können die Folgen schwerwiegend sein:

1. Wenn man sich darauf konzentriert, vorgegebene Anforderungen umzusetzen, und nicht genügend darauf achtet, wie gut jedes fertige Inkrement tatsächlich den Bedürfnissen entspricht, kommt es fast zwangsläufig zu mittelmäßigen Ergebnissen.
2. Ein Lernprozess auf Kosten der Auslieferung führt zu Produkten, die sich nicht selbst tragen. Wenn es auf Kosten der Infrastruktur geht, ist das Ergebnis Oberflächlichkeit und Zerbrechlichkeit. Es führt zu Produkten, die auf den ersten Blick vielleicht schön anzusehen sind, aber auf wackeligen Fundamenten stehen.

3. Eine übertriebene Betonung der Infrastruktur führt zu dem Ergebnis »nur Framework, kein Endprodukt«. Übermäßig ausgefeilte Systeme verzögern Interaktionen mit der realen Welt, bis sie nicht mehr relevant sind. Wenn die Infrastruktur zu wenig betont wird, bedeutet dies in der Regel, dass Probleme in die Zukunft verlagert werden. Die Anhäufung *technischer Schulden* wird eines Tages zurückgezahlt werden müssen.

Wie gut dies in der Praxis funktioniert, hängt natürlich stark von den beteiligten Personen, ihren jeweiligen Fähigkeiten und Verantwortlichkeiten und ihrem gegenseitigen Respekt ab. Bei Springboard DIY ist man sich der Konsequenzen einer unausgeglichenen Balance voll bewusst und arbeitet hart daran, es richtig zu machen. In ihrer Rolle zum Beispiel ist Alex eine starke Verfechterin für ihre Produkte, deren Benutzer und deren jeweilige Bedürfnisse. Gleichzeitig versteht sie ihre gegenseitige Abhängigkeit zu Kollegen in den Bereichen Wirtschaft, Lieferung, Technologie und Design sehr gut. Jeder im Führungsteam ist sich bewusst, dass er oder sie auf sich allein gestellt nichts Nennenswertes erreichen kann, und weiß aus Erfahrung, dass die wirklichen Innovationen oft in der Zusammenarbeit liegen. Alex hat dies oft genug mitgemacht, sodass sie darauf vertraut und gelernt hat, den Prozess zu begrüßen, so unklar die weiteren Schritte am Anfang auch erscheinen mögen. Egal, welche ihrer Kolleginnen und Kollegen Sie fragen, ihre Erfahrung wäre ähnlich.

2.2 Welche Art von Lean-Agile?

In Kapitel 1 haben wir einige der ganz unterschiedlichen Arten gesehen, wie Lean verstanden wird. Bevor wir zu Lean-Agile kommen, lassen Sie mich die Art von Agile, um die es in diesem Buch geht, beschreiben. Sie sollte bereits vertraut klingen:

Agile

Die Zusammenarbeit von Menschen an einer schnellen Entwicklung funktionierender Software, die bereits früh anfängt, Kunden einen Mehrwert zu bieten.

Ein bisschen ausführlicher:

Agile

Die Zusammenarbeit von Menschen mit verschiedenen Fähigkeiten an einer schnellen Entwicklung funktionierender Software, die bereits früh anfängt, Kunden einen Mehrwert zu bieten, und die in Teams erfolgt, die großen Wert auf Zusammenarbeit und Anpassungsfähigkeit legen.

Das ist meine stark gekürzte und »von rechts« ausgehende Interpretation des Agilen Manifests (*agilemanifesto.org*), die so formuliert ist, dass sie einen optimalen Punkt für digitale Ansätze beschreibt. Wenn Sie wissen wollen, was Agile ist, ist das Manifest der Ausgangspunkt. Agile ist kein definierter Prozess, eine Methode oder ein Framework; Agile bedeutet, dass die Werte des Manifests – Werte der Zusammenarbeit, funktionierende Software und Anpassungsfähigkeit – einbezogen werden.

Die agile Bewegung existiert, weil die Werte des Manifests bei vielen Menschen auf Resonanz gestoßen sind. Sie erleben, wie sie sich in sinnvollen Gesprächen entfalten können und die Möglichkeit haben, Dinge zu schaffen, die für die Menschen tatsächlich funktionieren, sowie die Fähigkeit, die Arbeitsumgebung zum gegenseitigen Nutzen von Entwicklern, Kunden und der Organisation ständig zu verbessern. Mit anderen Worten, sie sprechen einige menschliche Grundbedürfnisse an, die nicht jeder Arbeitgeber befriedigt.

Damit ein Wertesystem jedoch mehr als nur Wunschdenken ist, muss es eine klare Beziehung zwischen den Werten, den erwarteten Verhaltensweisen und den Annahmen, die diesen Verhaltensweisen zugrunde liegen, geben. Im Fall von Agile sind diese Annahmen gut dokumentiert – nicht zuletzt durch das Manifest selbst – und sie gehen weit über die Beschreibung der Verhaltensweisen hinaus:

- **Annahme 1**
  Eine direkte und kontinuierliche Zusammenarbeit mit den Kunden ist notwendig, um ein gegenseitiges Verständnis der Bedürfnisse und potenziellen Lösungen zu entwickeln und aufrechtzuerhalten. In komplexen Umgebungen ist zu erwarten, dass sich beide im Laufe der Zeit weiterentwickeln, während sie einander beeinflussen.
- **Annahme 2**
  Die Kooperation zwischen Menschen, die über den gesamten Prozess hinweg zusammenarbeiten, macht das große Ganze stärker als die Summe seiner Teile. Es ergeben sich nicht nur mehrere Perspektiven, die bei komplexen Problemen zum Tragen kommen, sondern auch neue Ideen, die in den Interaktionen entstehen und verfeinert werden.
- **Annahme 3**
  Der effektivste Weg, etwas in einer komplexen Umgebung aufzubauen, besteht darin, mit etwas zu beginnen, das bereits funktioniert, und dafür zu sorgen, dass es auch so bleibt, während es sich weiterentwickelt [31]. Dies gilt nicht nur für Produkte, sondern auch für die Arbeitsumgebung in Bezug auf ihre technische Infrastruktur, Prozesse, Praktiken, Organisation, Kultur und externen Beziehungen.

Der Durchbruch von Agile ist das Ergebnis der Zusammenführung dieser Annahmen, die in Form einer überzeugenden Wertevorstellung zur Geltung gebracht wurden. Die zugrunde liegende Botschaft ist klar: Wo immer diese Annahmen vermutlich zutreffen werden, ist es sinnvoll, sich entsprechend zu verhalten – auch wenn dies eine radikale Abwendung von bisherigen Arbeitsweisen erfordert.

Die Vorstellung, dass Kundenbedürfnisse durch kleine Schritte und häufige Iterationen erfüllt werden, deutet etwas Flow-Ähnliches an. Tatsächlich ist ein populärer Teil des Agile-Jargons »*flow of value*«[1]. Bedeutet dies dann, dass Agile nichts anderes ist als »*das strategische Streben nach Flow*«, also nichts anderes als das Lean aus Kapitel 1? Diese kontroverse Frage haben Fachleute, die ich schätze, sowohl positiv als auch negativ beantwortet:

- *Ja* – wie sollte es anders sein?
- *Nein* – Agile misst einigen Dingen mehr Wert zu als dem Flow (mit guter oder schlechter Auswirkung, je nach Standpunkt)

Eine Neudefinition von Agilität mit Begriffen aus Lean würde diese Frage nicht nur verfehlen, viele würden sie auch als eine Unverschämtheit betrachten. Stattdessen haben wir mit Lean-Agile eine Möglichkeit, ein gemeinsames Erbe zu würdigen. Es könnte wie folgt definiert werden:

Lean-Agile

Das strategische Streben nach Flow in komplexen Umgebungen, wobei die Organisation großen Wert auf Zusammenarbeit, kontinuierliche Auslieferung, Anpassungsfähigkeit und Lernen legt.

Es lohnt sich, diese Definition mit meinen beiden früheren Definitionen für »Lean« und »Agile« zu vergleichen. Die erste ist in Kapitel 1 aufgeführt, die zweite am Anfang dieses Abschnitts:

Lean

Das Streben nach Flow als strategische Notwendigkeit. Ein nicht endendes und wirkungsgerichtetes Vorhaben, das Mitarbeiter auf jeder Ebene der Organisation kontinuierlich und bewusst in einen Lernprozess einbindet.

1. Anm. d. Übers.: dt.: beständiger Wertfluss.

Agile

Die Zusammenarbeit von Menschen mit verschiedenen Fähigkeiten an einer schnellen Entwicklung funktionierender Software, die bereits früh anfängt, Kunden einen Mehrwert zu bieten, und die in Teams erfolgt, die großen Wert auf Zusammenarbeit und Anpassungsfähigkeit legen.

Manche Agilisten hätten keine Schwierigkeiten, alle drei Definitionen zu übernehmen und sie mit ihrem Verständnis von agilem Arbeiten in Einklang zu bringen. Andere würden sich bei dieser Sichtweise sträuben und die durchaus gerechtfertigte Position einnehmen, dass es bei Agile hauptsächlich um Teams oder hauptsächlich um Software geht. Von der Teamebene mal abgesehen, werden jedoch weitere Fragen aufgeworfen:

- Angesichts der Tatsache, dass Organisationen sowohl in komplexen Umgebungen eingebettet sind als auch solche aufweisen – Konkurrenzsituationen, Sozialsysteme usw. –, sollten sie nicht alle versuchen, in gewisser Weise agil zu sein?
- Ist es möglich, agiles Arbeiten nicht nur auf der Ebene der Entwicklungsteams, sondern auch auf der Ebene der Organisation zu ermöglichen? Mit anderen Worten, ist es möglich, Agile zu *skalieren*?

In diesem Buch geht es auch darum, diese Fragen ernst zu nehmen. Dazu beginnen wir mit einem funktionierenden System, das bereits anfängt, Kundenbedürfnisse zu erfüllen, der Digitalsparte von Springboard DIY. Wir untersuchen die unterschiedlichen Modelle, Methoden, Werkzeuge und Muster, die uns dabei helfen, dieses System einzurichten und seine kontinuierliche Weiterentwicklung zu unterstützen. Sie können selbst darüber urteilen, was auf diesem Weg gewonnen oder verloren gegangen sein könnte.

2.3 Schauen Sie nach

Kehren wir also zu Springboard DIY zurück und schauen wir uns an, wie ein schneller und reibungsloser Flow aussieht. Dabei werden wir seine Value Stream Map skizzieren.

Natürlich fangen wir von rechts an. Aber wo ist das genau? Um die Perspektive eines der Kunden von Springboard einzunehmen, könnten wir dies als den Zeitpunkt definieren, an dem eine Onlinetransaktion abgeschlossen ist. Leider beinhaltet diese Auswahl zwei Probleme:

- Kunden besuchen Springboard DIY online oft auch aus anderen Gründen als zum Einkaufen.
- Die Mitarbeiter von Springboard sind nicht bei jeder Transaktion beteiligt, was den Geschäftsprozess für unsere derzeitigen Zwecke uninteressant macht, auch wenn er aus Gründen der Kundenerfahrung und technischen Architektur von Interesse sein könnte.

Es gibt jedoch noch einen anderen wichtigen Wert, den digitale Services besonders gut erfassen können. Es liegt ein enormer Geschäftswert darin, zu verstehen, was Kunden wollen und wie sie am liebsten interagieren. Wenn Liefer- oder Entwicklungsprozesse mit einem expliziten Schritt einer *Validierung* enden, der dazu dient, den vorausgehenden Prozess zu informieren, nennen wir den Prozess *validiertes Lernen* (ein Begriff aus Lean Startup, das wir im nächsten Kapitel behandeln werden). Das Ergebnis der Validierung ist ein Strom von *Erkenntnissen*; wenn Erkenntnisse verinnerlicht und umgesetzt werden, ist das *Lernen* ein sehr wichtiges Resultat.

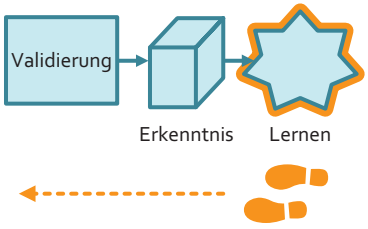

Abb. 2–1 *Validierung*

Das Commitment[2] von Springboard für validiertes Lernen erklärt das Gespräch zwischen Alex und Rowan am Anfang dieses Kapitels. Sie haben ein Experiment durchgeführt, bei dem sie einige Änderungen im Bereich der Elektrowerkzeuge auf ihrer Website erprobt haben. Mit einem Verfahren, das als A/B-Testing bezeichnet wird, wurden diese Änderungen nicht in einem Rutsch implementiert, sondern zunächst nur einem kleinen Teil der Benutzer zugänglich gemacht und dann erst mit wachsendem Vertrauen nach und nach eingeführt. Bei Experimenten wie diesen werden bestimmte Metriken im Voraus als Indikatoren für den Erfolg oder Misserfolg der Änderung festgelegt. Wenn man in der Größenordnung eines großen Einzelhändlers operiert, kann die statistische Aussagefähigkeit innerhalb weniger Stunden oder Tage erreicht werden. Die Veränderung kann anschließend dauerhaft implementiert werden, sofern sie als vorteilhaft beurteilt wird, oder andernfalls zurückgenommen werden.

2. Anm. d. Übers.: dt.: Selbstverpflichtung. Der Begriff »Commitment« ist in der Community allerdings so verbreitet, dass wir darauf verzichten, ihn im Laufe des Buches zu übersetzen.

Da das Team von Springboard schon seit Langem auf diese Weise arbeitet, werden viele Messungen vorgenommen, Sekunde für Sekunde, Benutzerinteraktion für Benutzerinteraktion. Da kommt eine Menge Daten zusammen, genug für alle! Produktspezialisten befassen sich mit Produkttrends und Finanzkennzahlen; Fachleute für User Experience (UX) untersuchen, wie (und wie schnell) sich die Benutzer durch die Website bewegen; technische Spezialisten befassen sich mit Leistungskennzahlen des Systems, um sicherzustellen, dass es sich wie erwartet verhält und in der Lage ist, Spitzen der Nachfrage zu bewältigen. Eine erfolgreiche Validierung hängt von der Kenntnis des aktuellen Ausgangsniveaus der relevanten Metriken und einigen expliziten Annahmen darüber ab, wie diese beeinflusst werden könnten. Oder um es anders auszudrücken: Eine gute Versuchsplanung ist die Voraussetzung für die Validierung.

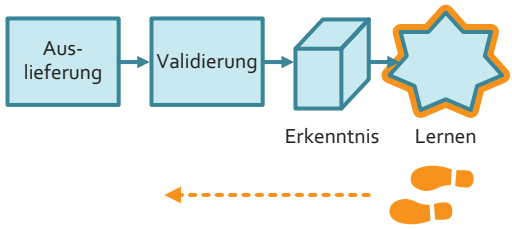

Abb. 2–2 *Auslieferung*

Um effektiv zu sein, erfordert validiertes Lernen schnelles Feedback. Damit ein schnelles Feedback praktisch und wirtschaftlich sinnvoll ist, muss es schnell, einfach und günstig sein, Änderungen einzuführen – die praktische Seite der kontinuierlichen Auslieferung. Glücklicherweise bedeutet schnell und einfach auch ein geringes Risiko, da der Umfang der Änderungen innerhalb eines Release klein genug ist, um allen Beteiligten bekannt zu sein, und der Auslieferungsprozess sich häufig wiederholt. Bei Springboard werden Auslieferungen nicht als besondere Ereignisse angesehen, sondern als »business as usual«. Mit Ausnahme von kritischen Zeiten mit hoher Kundennachfrage (z.B. Feiertagen mit verlängertem Wochenende) sind sie gerne bereit, mehrere Auslieferungen pro Tag durchzuführen, wenn sich die Gelegenheit dazu bietet.

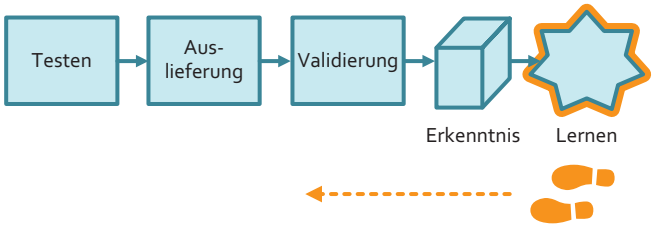

Abb. 2–3 *Testen*

Es versteht sich von selbst, dass das Digital-Team bei Springboard keine Änderungen implementiert, ohne sie vorher mehrfach überprüft zu haben. Allerdings verbringen die Qualitätsspezialisten den Großteil ihrer Zeit nicht mit den traditionellen Tätigkeiten des manuellen Testens neuer Funktionen oder der Koordination von Freigaben. Bis die meisten Änderungen bei ihnen eintreffen, sind die Funktions- und Integrationstests nicht nur abgeschlossen, sondern wurden so automatisiert, dass alle künftigen Rücknahmen sofort aufgefangen werden. Stattdessen arbeiten sie meist mit den Entwicklern an Änderungen, die sich noch in der Anfangsphase befinden, und führen Untersuchungen durch, messen die Systemleistung, testen im Auftrag von Benutzern mit speziellen Barriereanforderungen und so weiter – um sicherzustellen, dass alles, was die Onlineerfahrung betrifft, das gewohnte Aussehen und den nötigen Schliff hat, den die Kunden erwarten. Wenn sie nicht mit Kunden arbeiten, versuchen sie zumindest sich in sie hineinzuversetzen.

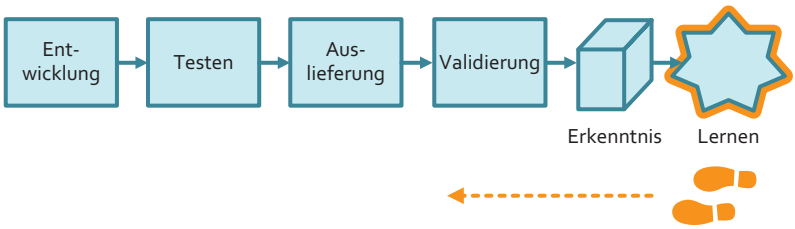

Abb. 2–4 *Entwicklung*

Die Entwicklung bei Springboard DIY erfolgt multidisziplinär und beinhaltet tiefgehende Spezialisierungen (serverseitige Infrastruktur, Benutzerschnittstellen und alles dazwischen), handwerkliches Können und ein hohes Maß an Zusammenarbeit. Kein Teammitglied arbeitet völlig unabhängig. Tatsächlich verbringen viele den größten Teil ihrer Zeit mit anderen Kollegen, etwa indem sie sich für längere Zeit mit einem anderen Teammitglied zusammentun (*pairing*[3]), bei besonderen Herausforderungen in größeren Gruppen zusammenkommen (*swarming*[3]) oder mit Kollegen aus anderen Teams zusammenarbeiten. Der Schwerpunkt ist »funktionierende Software«. Änderungen werden im Laufe des Tages integriert, wobei die Entwickler Rückmeldungen über ihre Arbeit erhalten, sowohl während sie sie durchführen (durch Tests, die sie parallel zu ihren Änderungen schreiben und während der Arbeit wiederholt ausführen) als auch während diese Änderungen mit Kollegen geteilt werden (durch Tests, die auf gemeinsam genutzten Servern laufen, und Integration mit anderen Services).

3. Anm. d. Übers.: Pairing und Swarming sind bekannte agile Praktiken, die Entwicklungsteams dabei helfen, technische Exzellenz zu erreichen.

All diese Arbeit wird im Hinblick auf den Moment erledigt, der normalerweise nicht mehr als ein paar Tage entfernt liegt, wenn sich das Ergebnis in den Händen der Kunden befindet und etwas Nützliches tut, also im wahrsten Sinne des Wortes »funktioniert«. Ein Teil der technischen Herausforderung – und auch der Chance der Digitalisierung – besteht darin, das Ganze so zu gestalten, dass das Team ein schnellstmögliches Feedback darüber erhält, wie gut jede Änderung aus Benutzersicht funktioniert.

Gute Datensätze bieten weitere Möglichkeiten, z. B. die Fähigkeit, personalisierte Empfehlungen auszusprechen, was bei guter Umsetzung den Nutzern zugutekommt. Die Daten müssen jedoch mit großer Sorgfalt verwaltet werden. Datenschutz und Informationssicherheit sind nicht nur auf technischer Ebene in das Onlineangebot bei Springboard integriert, sondern auch in den Richtlinien, die bestimmen, wie das Ganze konzipiert, aufgebaut und betrieben wird. Der Respekt vor Menschen geht über die Organisation hinaus!

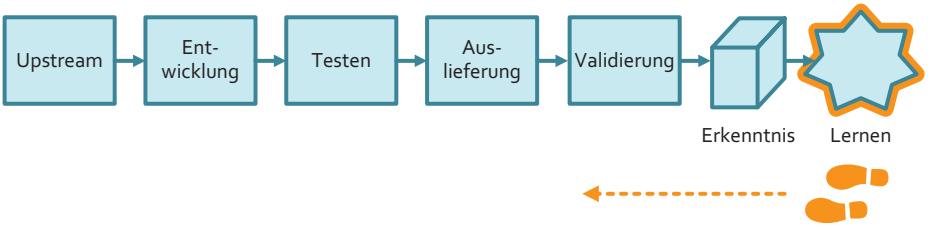

Abb. 2–5 *Upstream*

Ich habe »Benutzer« und »Kunde« teils austauschbar verwendet, aber es ist wichtig, sich vor Augen zu halten, dass die Benutzer eines Produkts nicht unbedingt diejenigen sind, die dafür bezahlen. Kunden, Sponsoren und andere Stakeholder sind für die Finanzierung, die Ziele und verschiedene organisatorische Richtlinien und Einschränkungen verantwortlich, können aber oft nur schlecht beurteilen, was für die Nutzer wirklich funktioniert. Bei Springboard DIY geben die Endkunden Geld über den Onlineshop aus und bezahlen nur indirekt für die Website selbst. Dennoch gibt sich Springboard keinen Illusionen hin: Nur wenn die Nutzer begeistert sind, werden sie die Website wieder besuchen. Dies wiederum bedeutet, dass die Entwickler eng mit User Researcher (Teil des Produktteams) und UX-Spezialisten (bei denen man bei Springboard davon ausgeht, dass sie sich sowohl im Produktlager als auch in der Technologie auskennen) zusammenarbeiten. Kurz gesagt, wann immer Entwickler keinen echten Kunden zur Verfügung haben, können sie auf etwas zurückgreifen, von dem sie hoffen, dass es das Nächstbeste ist.

Zur Förderung der Entwicklung gehört eine Reihe von Aktivitäten, die so miteinander verflochten sind, dass jede lineare Darstellung höchst irreführend wäre. Hier sehen wir User Research, Ideenfindung, Analyse, Prototypentwicklung, Design und so weiter – das sind alles Aktivitäten, die früher getrennt verwaltet wurden, heute jedoch noch als verschiedene Facetten eines Prozesses betrachtet werden, der nur ein Ziel hat: sicherzustellen, dass nachgelagerte Aktivitäten (alles zu ihrer Rechten) immer einen angemessenen Arbeitsaufwand mit dem höchstmöglichen Wert haben. Daraus sind Bezeichnungen wie *Fuzzy front end* [32] und *Upstream* (stromaufwärts/vorgelagert im Flow relativ zur Entwicklung) entstanden. Keiner der beiden Begriffe ist allgemein anerkannt; der Einfachheit halber verwende ich den letzteren, kürzeren.

Die Konzentration im Upstream auf den Nutzen erklärt die nicht lineare und vielschichtige Natur. Auf dem Weg zu diesem letzten Validierungsschritt hängt viel von der Hypothese und ihren Annahmen über die Benutzergruppe, deren Bedürfnisse, mögliche Änderungen des Benutzerverhaltens, technischen Herausforderungen und weiteren Faktoren ab. Viele dieser Annahmen können getestet werden, lange bevor ausgereifte Lösungen entwickelt wurden, wodurch die Organisation sehr viel schneller lernt und das Ergebnis der nachgelagerten Arbeit maximiert werden kann. Und es gibt keinen besseren Weg, Annahmen über Benutzer zu testen, als tatsächlich mit Menschen zu sprechen, sowohl direkt mit den Benutzern als auch mit denen, die regelmäßig mit ihnen arbeiten. Kundengruppen, Testlabore, Feldforschung und der Kunden-Helpdesk spielen alle eine Rolle. Alle bieten unterschiedliche Möglichkeiten, Ideen, Prototypen und Arbeiten in verschiedenen Phasen der Fertigstellung zu testen. Der Upstream hat daher das Potenzial, Menschen aller Fachrichtungen einzubeziehen, und das Team bei Springboard verfolgt eine Politik, die sicherstellt, dass jeder tatsächlich regelmäßig an die Reihe kommt.

Bei Springboard hat man gelernt, Upstream nicht mehr als eine in Etappen unterteilte Phase mit Übergängen dazwischen zu betrachten, sondern versteht es jetzt als die Verwaltung eines Portfolios von Optionen für die bestmögliche Rendite. Die besten Ideen werden schnell veröffentlicht, vielleicht sogar noch bevor sie vollständig ausgearbeitet sind – die Möglichkeit, die sich bietet, mag so bedeutsam sein, dass mehr Fachkräfte früher einbezogen werden sollten. Einige Ideen brauchen länger, um zu reifen, wobei die Kosten-Nutzen-Gleichung so marginal ist, dass einige Runden Prototyping und Benutzertests dem Team später unnötige Mühe ersparen könnten. Die am wenigsten vielversprechenden Ideen werden eine Weile ruhen, bevor eine Entscheidung getroffen wird, sie abzulehnen. Niemand trauert einer abgelehnten Option nach; jede einzelne ist eine Kugel, der ausgewichen wird – Verschwendung wird dadurch vermieden.

2.4 Ineffizienter Flow

Anstelle der *7 Verschwendungen* aus der Lean Production finden Sie hier eine einfache Klassifizierung von *Hindernissen für Flow,* die typischerweise bei digitaler Arbeit auftreten:

- Hindernisse, die sich auf einzelne *Arbeitspakete* auswirken – Arbeitspakete sind die Teile der Arbeit, die durch das System fließen –, von den Experimenten, Eigenschaften oder kleinen, aber dennoch lieferbaren »Scheiben« davon.
- Hindernisse, die sich auf die Mitarbeiter auswirken, die die Arbeit erledigen.
- Hindernisse, die sich auf das Gesamtsystem auswirken.

Innerhalb dieser drei groben Kategorien gibt es einige spezifische Hindernisse. Sie sind überall anzutreffen, und die meisten ihrer Bezeichnungen sind nicht nur in der Lean-Agile-Welt weithin anerkannt, sondern haben auch eine lange Vorgeschichte in der akademischen Literatur.

Wenn ein einzelnes Arbeitspaket am Flow gehindert wird, befindet es sich in der Regel in einem von zwei Status:

- **Blockiert**
  Ein Arbeitspaket gilt als blockiert, wenn es einen unzulässigen Status aufweist, der es am Fortschreiten hindert. Die unmittelbare Ursache wird als *Blocker* bezeichnet, und auch diese können klassifiziert werden: nicht behobene Fehler, ungelöste Abhängigkeiten, fehlende Informationen, Annahmen, die sich als ungültig erweisen, und so weiter.
- **Verzögert**
  Nicht zu verwechseln mit »blockiert werden«: Arbeitspakete sind verzögert, wenn im System keine Kapazität für ihre Bearbeitung zur Verfügung steht. Da die Aufmerksamkeit des Systems woanders liegt, bleibt dieses Hindernis oft unbemerkt.

Die nächsten beiden Hindernisse haben die unmittelbarsten (und schwerwiegendsten) Auswirkungen auf Menschen und Systeme:

- **Überlastung**
  Das haben wir bereits im ersten Kapitel gesehen: Menschen und Systeme sind überlastet und überbeansprucht, sie können aufgrund einer zu großen Nachfrage, die die Kapazitäten übersteigt, nicht ihr Bestes leisten. Zusätzlich zu den leidvollen Auswirkungen auf die Menschen hat die Überlastung verheerende Folgen für die Qualität und Vorhersehbarkeit, und Littles Gesetz gilt immer noch (egal wie sehr Sie es versuchen, gegen die Mathematik kommen Sie nicht an).

- **Unterforderung**
  So wie Menschen nicht gerne überlastet werden, mögen sie es auch nicht, wenn ihnen die Möglichkeit vorenthalten wird, einer sinnvollen Arbeit nachzugehen. Unterforderung tritt typischerweise im Anschluss an eine *Engpassaktivität* [33] auf, die sich nicht vorhersehbar verhält (vielleicht ist sie mit zu vielen Arbeitspaketen überlastet oder erstickt quasi an größeren Aktivitäten), oder die Engpassaktivität selbst ist unterfordert. Unterforderung ist besonders teuer, wenn sie an einem kritischen Ort auftritt.

Bevor wir zur letzten Gruppe von Hindernissen übergehen, beachten Sie einige wichtige Zusammenhänge zwischen den bereits erwähnten:

- Verzögerte Arbeitspakete und Überlastung sind zwei Seiten derselben Medaille, aus zwei verschiedenen Perspektiven betrachtet. Wenn Sie die eine sehen, machen Sie wahrscheinlich gerade Erfahrung mit der anderen.
- Überlastung und Unterforderung sind zwei gegensätzliche und unerwünschte Extreme, die darauf hinweisen, dass etwas aus dem Gleichgewicht geraten ist. Wenn einer der beiden Status systemweit auftritt, bedeutet dies in der Regel, dass die Nachfrage ineffektiv gemanagt wurde. Wenn die Status zwischen verschiedenen Teilen des Systems wechseln, deutet das wahrscheinlich darauf hin, dass man Verbesserungen in Bezug auf die Gleichmäßigkeit vornehmen muss, z. B. einen proaktiveren Umgang mit Blockern und lernen, wie man sie verhindern kann, die richtige Größe von Arbeitspaketen, in Automatisierung oder in eine breite Weiterbildung investieren.

Unsere letzten drei Hindernisse wirken sich auf das Gesamtsystem aus:

- **Mängel**
  Genau wie in der materiellen Fertigung sind Mängel in der Produktentwicklung ein doppeltes, dreifaches oder vierfaches Ärgernis! Energie wird zunächst für die Entwicklung fehlerhafter Arbeit, dann auf das Erkennen und Beheben des Problems und schließlich auf den Umgang mit eventuellen Folgeerscheinungen aufgewendet. Je früher die Mängel gefunden werden, desto besser: Mängel, die lokal von einem Entwickler gefunden werden, der Tests an neuem und noch nicht freigegebenem Code durchführt, sind damit nicht gemeint. Die fehlgeschlagenen Tests werden eher als Platzhalter für noch zu leistende Arbeit angesehen. Das andere Extrem sind Mängel, die es bis »in die Produktion geschafft haben« und deren unerwünschte Auswirkungen auf die Kunden bewältigt und vielleicht sogar kompensiert werden müssen.
- **Nachfrage aufgrund von Fehlleistungen**
  Dieses Konzept hat seinen Ursprung in der Dienstleistung im öffentlichen Sektor [34]. Nachfrage aufgrund von Fehlleistungen bedeutet nicht notwendiger-

weise Mängel im herkömmlichen Sinne, dass die Arbeit nicht wie spezifiziert durchgeführt wurde, sondern dass die Arbeit den Kundenbedürfnissen so unzureichend entsprach, sodass der gesamte Prozess noch einmal durchlaufen werden muss, was zu zusätzlicher Nachfrage führt. Oberflächlich betrachtet mögen die Nachfrage aufgrund von Fehlleistungen und fehlgeschlagene Experimente ähnlich erscheinen, aber die Experimente sind so angelegt, dass daraus effizient gelernt werden kann, wenn die Ergebnisse ungewiss sind, während die Nachfrage aufgrund von Fehlleistungen meist vermeidbare Verschwendung bedeutet.

- **Ungenutzte Gelegenheiten**
  Damit ist die Verschwendung unter wirtschaftlichen Gesichtspunkten gemeint, die durch schlechte Entscheidungen bei der Dimensionierung und Reihenfolge der Arbeit entsteht. Beispiele hierfür sind wertvolle Arbeit, die hinter weniger wertvoller Arbeit in der Warteschleife steht (vielleicht, weil letztere ein Teil eines großen Auftrags ist), oder Ungleichgewichte zwischen verschiedenen Arbeitstypen, Bezugsquellen oder *Serviceklassen* (z.B. mangelnde Differenzierung zwischen terminabhängiger Arbeit, dringlicher Arbeit und langfristigen Wartungs- und Verbesserungsarbeiten). Kein Team verfügt über sämtliche Informationen, aber die erfolgreichsten Teams verstehen die wirtschaftliche Grundlage ihrer Arbeit im Ganzen, haben ein gutes Gespür für die voraussichtlichen *Verzögerungskosten (Cost of Delay)* [35], die auf einzelne Arbeitspakete zurückzuführen sind, und sind daher in der Lage, vor Ort gute Entscheidungen zu treffen.

Betrachtet man diese Hindernisse als Ganzes, so sollte klar geworden sein, dass an Flow mehr dran ist, als Menschen nur auszulasten. »Auslastung« ist vielleicht sogar ein schlechtes Zeichen! Lean-Agile bedeutet erstens, den Flow im Sinne des Kundennutzens zu steuern, und zweitens, das Richtige für die Menschen im System zu tun – beides zur gleichen Zeit.

2.5 Den Flow managen – von rechts nach links

Noch ohne Hilfe gewisser Tools und Frameworks, die im nächsten Kapitel vorgestellt werden, zeigt sich in den folgenden Fragen eine »Von rechts nach links«-Strategie zur Steuerung von Flow. Unter Berücksichtigung der Value Stream Map werden die Fragen in eine Abfolge von rechts nach links gestellt:

- Was lernen wir aus unseren Arbeitssystemen, sowohl aus den Veränderungen, die in letzter Zeit eingeführt wurden, als auch aus dem, was bereits seit einiger Zeit gebräuchlich ist?

- Was hoffen wir demnächst zu lernen und sind unsere Systeme und Experimente so angelegt, dass sie die Informationen liefern, die wir brauchen?
- Was ist bereit zur Auslieferung? Wann und wie wird es ausgeliefert?
- Was muss getestet werden? Wie, von wem, in welchem Umfeld und nach welchen Kriterien?
- Was wird derzeit entwickelt? Wer arbeitet woran? Was tun wir, um unsere Hindernisse zu beseitigen?
- Sind wir uns darüber im Klaren, was als Nächstes kommt? (Diese letzte Frage kann entfallen, wenn abzusehen ist, dass keine Kapazität für weitere Arbeit frei wird.)

Diese Fragen sind einfach – Teams, Manager und Coaches stellen sie aus Gewohnheit. Ein scharfsinniger Manager oder Coach konzentriert sich auf die wesentlichen Fragen; ein erfahrenes Team wird sich diese Fragen bereits selbst gestellt haben. Beginnen Sie im Zweifelsfall mit Fragen, die dazu beitragen, dass die Arbeit nicht nur durch das System, sondern auch aus dem System heraus fließt. *»Stop Starting, Start Finishing!«*

2.6 Den Flow optimieren – von rechts nach links

Damit auch nur ein einziges Arbeitspaket reibungslos durch das System fließen kann, müssen viele Dinge richtig laufen! Die obigen Fragen sollen einige der wichtigsten Ursachen ausschließen, dass etwas schiefgehen kann:

- Es wurde etwas ausgeliefert, aber nichts wurde dabei gelernt, entweder weil das Ergebnis nicht genutzt wird oder weil es so unzulänglich ausgeführt ist, dass uns benötigte Daten fehlen.
- Wir sind bereit zur Auslieferung, aber wir wissen nicht, was wir zu lernen hoffen (wir haben keine klare Hypothese zu prüfen), oder wir werden keine ausreichenden Nachweise erbringen.
- Es steht nichts bereit zur Auslieferung und wir werden auch absehbar nichts ausliefern können; alternativ haben wir etwas, das wir gerne ausliefern würden, was aber aus Gründen von Kompetenzen, Verfügbarkeit oder Zuständigkeit verhindert wird.
- Wir haben jetzt und auch in naher Zukunft nichts zu testen; wir sind aus irgendeinem Grund nicht bereit dazu oder Mängel blockieren uns.
- Arbeit ist in der Entwicklung blockiert, Arbeit, die vielleicht gar nicht erst hätte begonnen werden sollen.
- Es befindet sich gerade keine werthaltige Arbeit in der Entwicklung oder in der Pipeline.

Wie sollte eine Führungskraft in solchen Situationen reagieren? Eine gute Vorgehensweise funktioniert auf drei Ebenen:

1. Der Umgang mit dem unmittelbar anstehenden Problem, seinen Auswirkungen und seinen direkten Ursachen
2. Der Umgang mit systemischen Ursachen – die Tatsache, dass das System es zulässt, dass diese Probleme auftreten
3. Der Umgang mit der Tatsache, dass dieser Mangel im System nicht vorhergesehen wurde, bevor man diese Probleme hat entstehen lassen

Die Führungsverantwortung nimmt von Ebene zu Ebene zu. Ein Team, das in der Lage ist, mit seinen unmittelbaren Problemen und deren direkten Ursachen umzugehen, sollte dies auch tun dürfen und falls es dies noch nicht kann, in dieser Grundfähigkeit gecoacht werden. Wie könnten wir als Nächstes das System verbessern, um ähnliche Probleme weniger wahrscheinlich oder weniger schwerwiegend zu machen? Und zu guter Letzt, warum reagieren wir erst jetzt darauf? Haben wir es versäumt, vorauszuschauen, waren wir zu reaktiv oder sogar selbstgefällig?

2.7 Beschuldigen Sie das System und übernehmen Sie Verantwortung

Sie haben es zweifellos schon einmal gehört: »Geben Sie nicht der Person, sondern dem System die Schuld!« W. Edwards Deming nahm diesen Rat bekanntermaßen auf und gab Zahlen dafür an:

> *94 % der Probleme oder Mängel in einer Organisation werden durch »das System« verursacht.*

Deming sprach als Statistiker; seiner Meinung nach waren die verbleibenden 6 %, die den Menschen zugeschrieben wurden, »fast unerheblich«. Manchmal erhöhte er den Anteil des Systems an der Verantwortung auf 95 %; jetzt sei der Anteil der Menschen »statistisch unbedeutend«.

Um diesen allgemeinen Ratschlag zu konkretisieren und die natürlichen Instinkte zu zügeln, Schuldzuweisungen in unnütze Richtungen zu lenken, empfiehlt sich folgende Standardreaktion auf ein Scheitern:

- Bis das Gegenteil bewiesen ist, sind die meisten Misserfolge ein Versagen in der Zusammenarbeit.

Mit anderen Worten: Hätten die richtigen Gespräche zur richtigen Zeit stattgefunden oder hätten sich die Menschen für eine andere Weise der Zusammenarbeit entschieden, wäre ein Scheitern vermieden worden. Gute Prozesse fördern und

verstärken diese verbesserten Verhaltensweisen so lange, bis sie sich als kulturelle Normen so gut etabliert haben, dass alle Abweichungen auffallen und das System auf natürliche Weise selbstkorrigierend wirkt.

Nicht jeder Misserfolg lässt sich auf diese Weise erklären (z. B. Scheitern von Vorausplanungen oder Investitionen, beide Fälle gehören zur Führungsverantwortung), aber es ist eine wahrscheinlichere Annahme als die meisten Alternativen.

Lassen Sie mich ein paar persönliche Beispiele nennen:

1. Codereviews waren einst eine häufige Ursache für lange Verzögerungen und große Frustration in einer von mir geleiteten internationalen Abteilung. Hätten der Programmierer und der Reviewer des fraglichen Codes vor und während der Entwicklung angemessen zusammengearbeitet, hätte das Codereview auf jeden Fall weniger Überraschungen bereitgehalten. Hätten sie eng genug zusammengearbeitet (und vielleicht *Pair Programming* praktiziert, wie es mein nächstes Team tat), wäre der formale Reviewschritt vielleicht gar nicht notwendig gewesen! Sobald wir dieses Problem als ein Scheitern der Zusammenarbeit erkannten, folgte schnell eine Verbesserung.
2. Einige Jahre später stellte ich in meiner Interimsfunktion als Auslieferungsmanager fest, dass die Arbeit häufig in der Testabteilung ankam, ohne dass es einen klaren Plan für das weitere Vorgehen gab. Schlimmer noch, als die Tests schließlich begannen, wusste niemand, wie lange es dauern würde. Die bemerkenswert effektive Lösung bestand in einem kurzen Meeting[4], bei dem Vertreter von den Abteilungen Produkt, Entwicklung und Test unmittelbar vor Beginn der Entwicklung ein kurzes Gespräch über jeden Teil der Arbeit führen. Wir haben damit nicht nur diese frustrierenden Verzögerungen beim Test abgewendet, sondern auch die Bedingungen für die Zusammenarbeit während des gesamten Prozesses geschaffen.

Änderungen wie diese sind schnell und einfach zu vereinbaren, erfordern kein großes Fachwissen und ihre Vorteile überwiegen bei Weitem die Kosten. Natürlich ist nicht jedes Problem ganz so einfach zu lösen, geschweige denn zu verhindern, bevor es überhaupt entsteht, und einige erfordern erhebliche Investitionen. Glücklicherweise treten in der digitalen Welt viele Probleme des Flow so häufig auf, dass es bereits viele gut dokumentierte Maßnahmen zu ihrer Bewältigung gibt. Im nächsten Kapitel werden wir einige der wichtigsten Frameworks und ihre Kombinationsmöglichkeiten behandeln.

4. Anm. d. Übers.: Ein solches Meeting wird in der agilen Community auch »Three Amigos Conversation« genannt.

2.8 Fragen zum Kapitel

1. Wie fördert Ihre Produktentwicklungsorganisation die Zusammenarbeit, sowohl intern (innerhalb und zwischen Teams) als auch extern (vor allem mit Kunden)?
2. Was würde Ihre Produktentwicklungsorganisation unter dem Begriff *funktionsfähige Software* (oder allgemeiner: *funktionsfähiges Produkt*) verstehen? Wie kompatibel ist dieses Verständnis mit dem Konzept der *kontinuierlichen Lieferung*?
3. Wie passt sich Ihre Produktentwicklungsorganisation an das sich stetig ändernde Wissen an – produktbezogen, prozessbezogen und organisatorisch? Wie fördert und erfasst sie dieses Wissen?
4. Wie sorgt Ihre Produktentwicklungsorganisation für eine angemessene Balance der Aufmerksamkeit zwischen Auslieferung, Entwicklung und Infrastruktur (letztere umfasst Technologie, Prozess und Kultur)?
5. Wenn Sie von rechts nach links arbeiten, wie verstehen Sie den Wertstrom Ihrer Produktentwicklung?
6. Inwieweit ist der Wertstrom Ihrer Produktentwicklung rechts in der Validierung verankert? Welche Formen nimmt diese Validierung an?
7. Welche »vorgelagerten« Aktivitäten versorgen den Produktentwicklungsprozess mit hochwertiger Arbeit? Wie schaffen es die besten Ideen an die Spitze der Warteschleife?
8. Wie managen Sie die Arbeit aus Ihrem Produktentwicklungsprozess heraus (im Gegensatz zu hinein)?
9. Wie erkennen, mindern und beheben Sie *»ineffizienten Flow«*: blockierte Arbeit, verzögerte Arbeit, Menschen, Teams oder Systeme, die entweder überlastet sind oder denen es an hochwertiger Arbeit mangelt, Mängel, Nachfrage aufgrund von Fehlleistungen, ungenutzte Gelegenheiten?
10. Wenn Sie von rechts nach links entlang Ihres heutigen Wertstroms der Produktentwicklung arbeiten, welche der oben genannten Hindernisse für Flow würden Sie als Erstes erwarten? Wenn Sie die Übung wiederholen würden, nachdem Sie diese Hindernisse angesprochen haben, was würden Sie vorfinden?
11. Wie viele Ihrer wiederkehrenden Hindernisse können als *»Versagen in der Zusammenarbeit«* bezeichnet werden? Wie hilft diese Einordnung?