Dellnitz · Gentsch · Demarmels · Sierralta · Vigenschow

FOKUS

Das Handbuch für Product Owner

+ Scrum für moderne und agile Produktentwicklung nutzen
+ Vom Product Backlog bis zum Krisen- und Konfliktmanagement
+ Randvoll mit lebendiger Theorie und handfesten Methoden

Rheinwerk Computing

Kapitel 3
Das Fundament: Projektmanagement

*Agil ist nicht gleich agil. Jedes Vorhaben hat seine Eigenheit, und deswegen brauchst du als Product Owner*in deine individuelle, passende Strategie, wie du bei der Entwicklung deines Produkts vorgehen möchtest.*

»Ich muss das noch anders in Form bringen.« Auf Ellens Schreibtisch liegt ein Bogen Flipchart-Papier mit einer riesigen Mindmap. An den Wänden kleben Post-its in allen Formen und Farben. Auf dem Monitor flimmert ein buntes Analytics-Dashboard. In Ellens Kopf ist alles klar, sie weiß genau, wie das Produkt erfolgreich sein wird. Jetzt muss sie überlegen, wie die einzelnen Aspekte zusammenhängen: Was kommt zuerst? Und was später? Wie lässt sich schnell Nutzen stiften? Und wie kann sie Risiken in den Griff kriegen? Sie braucht einen groben Plan, eine Übersicht, bevor sie sagen kann, was sie im Product Backlog nach oben zieht. Aber ist das agil? Ein Plan? Ellen kaut auf ihrem Bleistift. »Egal«, denkt sie, »ich mache das einfach.«

Ellen hat es gut erfasst: Eine zentrale Aufgabe in deiner Rolle als Product Owner besteht darin, eine gute Strategie zu entwickeln, wie du mit deinem Team vorgehen willst, welche Themen ihr früher oder später bearbeitet und in welcher Tiefe ihr sie bearbeiten wollt. Hier hilft gutes Projektmanagement weiter: Meilensteine identifizieren, Vorgehensstrategien entwerfen und frühzeitig kommunizieren. Deine *Vorgehensstrategie* muss zu deinem Produkt, den verwendeten Technologien, den Rahmenbedingungen sowie den Erfahrungen und Fähigkeiten deines Teams passen. Du ordnest dazu die Ergebnisse aus der Übersichtsphase, um sie in eine agile Planung zu übersetzen. Diese Planung sollte deine Vorgehensstrategie so klar widerspiegeln, dass es dir leichtfällt, dich mit deinem Umfeld und dem Team dazu abzustimmen und den Kurs zu setzen. Du bist dann bereit, dein wichtigstes Arbeitsmittel aufzusetzen und dein Product Backlog initial mit den ersten kleinen Entwicklungsschritten zu füllen, um deine Arbeits- und Planungshypothesen zu überprüfen.

In diesem Kapitel erfährst du, wie du aus Meilensteinen eine angemessene Vorgehensstrategie für dein Produkt ableitest, worauf es bei der Kommunikation in der Planungsphase ankommst und wie du die Besonderheiten deines Produkts bei der Planung berücksichtigen kannst.

3.1 Meilensteine, Iterationen und das Produkt-Ziel

Die Kombination aus Meilensteinen zur Definition von sinnvollen mittel- und langfristigen Planungszielen und Iterationen, um euch an diese heranzuarbeiten, hat sich aus unserer Sicht sehr bewährt. Der Scrum Guide enthält seit November 2020 mit dem neu eingeführten Konzept des *Produkt-Ziels* (engl. Product Goal) eine sehr ähnliche Idee. Wir zeigen dir, wie du diese Werkzeuge nutzt, um eine effektive Vorgehensstrategie zu entwickeln und zu kommunizieren.

3.1.1 Meilensteine

Iterationen, ja klar, aber *Meilensteine* in agilen Projekten? Meilensteine wirken auf viele Menschen wie Konzepte aus der »alten Planungswelt«. Das stimmt, denn häufig werden sie lediglich als Terminvorgaben missbraucht. Und gleichzeitig sind sie immer noch ein nützliches Konstrukt, wenn sie denn gut gemacht werden.

Ein Meilenstein ist ein wichtiger Entscheidungspunkt im Projekt (GPM, 2015). Und Entscheidungen müssen ja auch in agilen Projekten getroffen werden. Er definiert ein *inhaltliches* Ziel, an dem ihr innehaltet und prüft, wo ihr mit eurem Produkt steht und wie es weitergehen kann. Zu einem Meilenstein gehört auch ein grob geschätzter Termin. Er soll sicherstellen, dass ihr, auch wenn ihr eure Ziele nicht zu dem geplanten Zeitpunkt erreicht, ebenfalls innehaltet und schaut, was das bedeutet. Es ist nicht automatisch schlimm, wenn ihr nicht rechtzeitig ankommt. Schließlich läuft nicht immer (eigentlich sehr selten in der Projektarbeit) alles nach Fahrplan, und im Fall der Fälle könnt ihr immer entscheiden, den Meilenstein zu verschieben.

Allerdings sehen leider viele Meilensteinpläne traditionell ungefähr so aus:

1. Fachkonzepte fertig, Start der Entwicklung.
2. Halbzeit des Projekts, wir müssen mal irgendwas zeigen, damit man uns glaubt, dass wir vorankommen.
3. Jetzt sind wir fast fertig, wir sollten noch mal was zeigen und überlegen, wie wir die Einführung machen.
4. Beginn der Testphase, wir entwickeln nichts Neues mehr.
5. Roll-out und Projektabschluss.

Derartig generische Meilensteinpläne sehen auf den ersten Blick nicht verkehrt aus, und wir haben schon Projekte für einige Millionen gesehen, die auf so einer Basis freigege-

ben und gestartet wurden. Allerdings sagen solche Pläne gar nichts darüber aus, was das Projekt eigentlich leisten will/muss, außer irgendwann irgendwomit fertig zu werden.

Du kannst das besser – mit einem *Meilensteinplan*, der tatsächlich etwas über dein Projekt und die Herausforderungen aussagt, die du erwartest. Der Leitgedanke eines guten Meilensteinplans ist, dass er die Strategie deines Vorgehens erzählt:

- Was glaubst du: Welchen Herausforderungen werdet ihr euch bei der Entwicklung stellen müssen?
- Was wollt ihr daher tun, und in welcher Reihenfolge wollt ihr vorgehen?

In einem derartigen Meilensteinplan liegt der Fokus nicht auf den Terminen, sondern im logischen Aufbau eures Vorgehens.

Agile Meilensteine sind keine zu erfüllenden Featuresets, sondern spiegeln die zentralen zu klärenden Fragen wider. Dadurch lassen sie sich in der Regel bereits sehr früh in der Entwicklung erkennen und beschreiben.

Meilensteine sind fest

Der Kern eines Meilensteins – die zu treffenden Entscheidungen und die dazu zu klärenden Fragen – ist erst mal fest. Wenn der Meilenstein nicht in der geplanten Zeit erreicht wird, wird er verschoben. Die dafür zu realisierenden Inhalte können nach Bedarf angepasst werden.

Wenn ihr einen Meilenstein beschreibt, sollten die folgenden Elemente auftauchen:

- Eine wichtige Frage an euer Produkt – so etwas wie:
  - Will/braucht unsere Kundschaft das (Feature) so?
  - Ist unsere Lösung verständlich und nachvollziehbar für die Nutzerschaft?
  - Reicht die Performance? Müssen wir neue Geräte kaufen?
  - Wie aufwändig wird es wirklich, die Anbindung herzustellen?

  Du kannst diese Fragen aus der Risikoliste ableiten, die du in der Übersichtsphase begonnen hast (siehe dazu auch Kapitel 2, »Alles im Blick: die Produktübersicht«).
- Welche Ausbaustufe des Produkts plant ihr bis zu diesem Punkt zu entwickeln, die euch hilft, diese Frage ganz praktisch zu beantworten? Was vom Produkt wird da sein, was auch nicht, weil es nicht zwingend notwendig ist?

Zu guter Letzt versuche, einen guten, möglichst sprechenden Titel für den Meilenstein zu finden.

Aus unserem Erfahrungsschatz: Der »Einfach kopieren«-Meilenstein

Vor einigen Jahren haben wir in einem Projekt den »Einfach kopieren«-Meilenstein definiert. Der Hintergrund war, dass der Projektauftrag darin bestand, ein aufgekauftes nationales Softwareprodukt »einfach in die bestehende Systemlandschaft des Konzerns zu kopieren« und dabei

- die verwendete Programmiersprache und die Nutzeroberflächen von C# nach Java zu portieren,
- die internationale und länderübergreifende Verwendung an allen Standorten des Konzerns zu ermöglichen,
- die hauseigenen Buchhaltungs- und Verwaltungssysteme anzuschließen.

Das Team hatte als ein zentrales Risiko identifiziert, dass aufseiten des Managements kein ausreichendes Verständnis für die Komplexität und die Konsequenzen aus dem Wechsel der Programmierplattform sowie der Internationalisierung des Produkts bestand.

Den »Einfach kopieren«-Meilenstein haben wir daher grob wie folgt beschrieben:

1. Entwicklung der meistgenutzten Auftragsmaske, um Folgendes abzusichern:
   - Erreichen wir dieselbe Qualität und Geschwindigkeit bei der Eingabe wie im gekauften System – insbesondere im Hinblick auf die Keyboard-Navigation?
   - Wie verändern sich der Platzbedarf und das Layout durch die Internationalisierung im Hinblick auf zusätzlich zu erfassende Informationen und unterschiedliche Textlängen in den unterschiedlichen Sprachen?
   - Müssen gegebenenfalls an einigen Standorten größere Monitore angeschafft werden, um eine effiziente, ergonomische Bedienung zu ermöglichen?
   - Welche Konsequenzen lassen sich aus der neuen länderübergreifenden Verwendung (beispielsweise Zeitzonen, Zollregelungen) an diesem Beispiel bereits erkennen?
   - Was ergibt sich aus dem oben Gelernten für den weiteren Projektverlauf?
2. Bei der Umsetzung der Auftragsmaske fokussieren wir uns
   - auf die Oberflächen und den Eingabe-Flow, es erfolgt keine echte Verarbeitung der Daten im Backend,
   - auf ausgewählte Fehlerfälle bei der Eingabe, um die Anzeige von Fehlern und die Navigation zur Korrektur zu testen, nicht auf eine Abdeckung aller möglichen Konstellationen,
   - auf zwei bis drei ausgewählte Sprachen, nicht auf alle umzusetzenden.

Wenn du dir die Beschreibung des Meilensteins hier im Beispiel anschaust, fällt dir vielleicht auf, dass er positiv formuliert ist. Er stellt dar, was wir planen zu tun, um Erkenntnisse zu generieren und Risiken abzusichern (ohne dass wir die Risiken als solche benannt hätten). Du signalisierst auf diese Art deinen Stakeholder*innen, dass euch klar ist, wo es Probleme geben könnte (und eure Stakeholder*innen wissen es jetzt auch), aber dass ihr die Dinge im Griff habt und einen Plan, wie ihr damit umgehen wollt. Das ist enorm vertrauensbildend.

3.1.2 Sprints (Iterationen)

Sprints (beziehungsweise *Iterationen* in der agilen Begriffswelt außerhalb von Scrum) gehören zu den zentralen Konzepten der agilen Entwicklungsarbeit. Sprints teilen eure Arbeit in Zeitabschnitte fester Länge auf, die immer nach dem gleichen Muster strukturiert sind (siehe Abbildung 3.1).

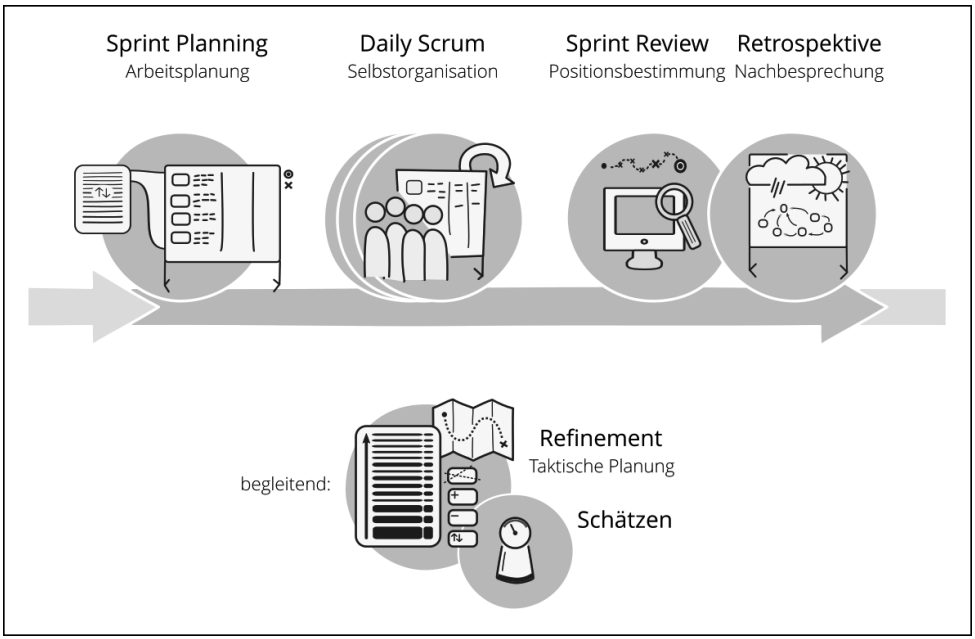

Abbildung 3.1 Die typische Struktur eines Sprints

Zum Start eines Sprints fokussiert ihr euch als Team auf ein (Zwischen-)Ziel, das ihr realistisch mit dem nächsten Sprint erreichen könnt. Das Entwicklungsteam prüft und plant dann gemeinsam, was dazu an Arbeit zu erledigen ist. Während des Sprints organisiert das Entwicklungsteam selbstständig, wie es die Arbeit erledigt. Du kannst dich in

der Zeit auf die Vorbereitung des nächsten Sprints konzentrieren. Für Rückfragen oder falls das Team deine Hilfe bei Entscheidungen braucht, solltest du aber stets greifbar sein. Am Ende des Sprints hältst du mit dem Team und ausgewählten Stakeholder*innen kurz inne, ihr schaut euch den erreichten Ausbaustand eures Produkts an und prüft, wie es inhaltlich weitergehen soll.

Wenn ihr am Ende eines Sprints das gesetzte inhaltliche Ziel nicht ganz erreicht habt, prüft ihr, was das bedeutet, und plant die Inhalte um. Umplanen kann bedeuten, dass ihr die fehlende Funktionalität erst mal parkt und mit anderen Sachen weitermacht oder dass ihr direkt in der nächsten Iteration versucht, die Aufgaben abzuschließen. Mehr dazu findest du in Kapitel 12, »Kurs anpassen: das Review«.

Die Länge eines Sprints sollte zwischen zwei und vier Wochen liegen. Tatsächlich haben sich in der Praxis drei Wochen sehr bewährt. Drei Wochen sind ein Zeitraum, in dem das Team genügend Zeit hat, in Ruhe zu arbeiten und auch größere Aufgaben zu bewältigen. Gleichzeitig ist der Zeitraum noch so kurz, dass er überschaubar ist und ihr »einfach machen« könnt, ohne euch zu verlaufen.

Das Iterationsraster

Bei Iterationen ist die Zeitdauer fest. Dadurch entsteht ein Iterationsraster aus immer gleich ablaufenden Intervallen fester Länge. Das Raster erfüllt zwei Funktionen:

- Ihr werdet gezwungen, euch die Arbeit in kleine, überschaubare Schritte aufzuteilen, die in das Raster passen. Das sorgt für Fokussierung und erleichtert die Planung.
- Das Iterationsraster ist auch ein Prüfraster, um Probleme frühzeitig zu erkennen und ein realistisches Bild von eurer Geschwindigkeit zu entwickeln.

Wenn am Ende einer Iteration Inhalte nicht (fertig) umgesetzt werden konnten, wird geprüft, wann und gegebenenfalls ob diese noch ungesetzt werden.

3.1.3 Meilensteine und Iterationen kombinieren

Das klingt erst mal alles sehr ähnlich zu Meilensteinen, nur schlanker und agiler. Weshalb solltest du trotzdem beide Werkzeuge verwenden? Ein Meilenstein beschreibt ein größeres Ziel, auf das ihr als Team hinarbeitet und das von strategischer Bedeutung ist. Sie sind die Leuchtfeuer, zu denen ihr mithilfe einer Reihe von Iterationen navigiert. Mit jeder Iteration prüft ihr, ob ihr noch auf Kurs seid, wie gut ihr vorankommt, und könnt gegebenenfalls nachsteuern (siehe Abbildung 3.2).

Auf dem Weg zu einem Meilenstein liegt also eine Reihe von Iterationen, in denen ihr euch um das Thema des Meilensteins kümmert. Durch die Einbindung von Stakehol-

der*innen in die Reviews ist bereits unterwegs klar, wie es um den Meilenstein bestellt ist, und die notwendigen Entscheidungen können in Ruhe vorbereitet werden.

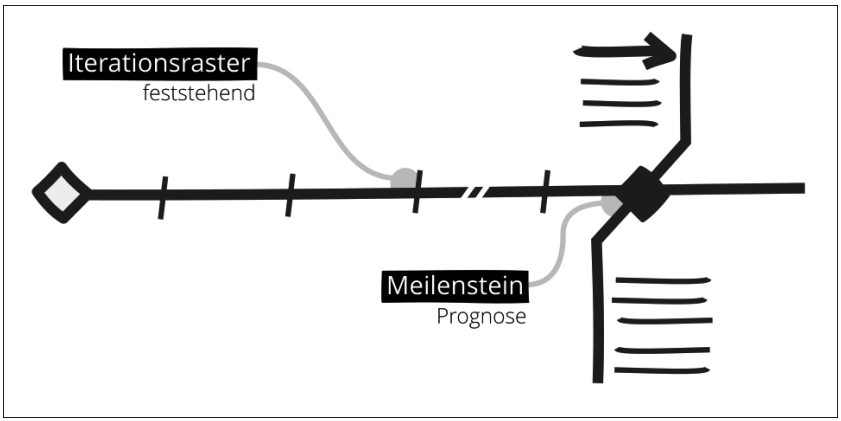

Abbildung 3.2 Das Iterationsraster steht fest, aber die Anzahl der Iterationen nicht.

3.1.4 Produkt-Ziel oder Meilensteine, wo ist der Unterschied?

Im Scrum Guide gibt es seit dem November 2020 das Konzept des *Produkt-Ziels*. Es ist dort definiert als Beschreibung eines Ausbaustands des Produkts von Wert. Das Produkt-Ziel ist ein längerfristiges Planungsziel, wobei ein Scrum Team immer nur genau ein Produkt-Ziel verfolgen darf.

Wenn du unserer Vorgehensweise zur Identifikation und Beschreibung agiler Meilensteine folgst, entspricht der jeweils nächste anstehende Meilenstein einem Produkt-Ziel. In der Praxis gibt es keinen relevanten Unterschied. Die Wahl eines anderen Begriffs im Scrum Guide soll vermutlich vor allem unterstreichen, dass es um ein inhaltliches Ziel geht, während Meilensteine ja leider häufig als Terminziele missverstanden werden.

3.2 Deine Vorgehensstrategie entwickeln

Wie gehst du jetzt am besten vor, um sinnvolle Meilensteine zu finden und auf der Basis eine Vorgehensstrategie zu entwickeln? Du solltest aus der Übersichtsphase eine Liste mit Risiken und offenen Fragen haben. Bei einigen Punkten bist du vermutlich zuversichtlich, dass diese sich im Guten klären werden, andere bereiten dir mehr Sorgen.

Suche dir für den Start zwei bis drei der kniffligsten Punkte heraus, schaue sie dir genauer an und gehe die folgenden Schritte durch.

3.2.1 Woher kommen deine Sorgen?

Schaue dir die Punkte noch einmal genau an: Sind diese bereits so konkret beschrieben, dass dir klar ist, wo genau das Risiko herkommt? Falls nicht, versuche, dich in die Tiefe zu denken, bis du ein möglichst konkretes Problem benennen kannst.

Statt ...	Besser ...
Der Termin ist sportlich und kann möglicherweise nicht gehalten werden.	Die Anbindung an das Archivsystem ist noch unklar, sodass wir unter Umständen dafür mehr Zeit benötigen und den Termin nicht halten können.
Die Anbindung an das Archivsystem ist noch unklar.	Wir wissen noch nicht genau, in welchem Format und mit welcher Antwortzeit wir auf die Dokumente aus dem Archivsystem über die neue Schnittstelle zugreifen können. Das hat unter Umständen Auswirkungen auf die Ausgestaltung der Anwendungsfälle.
	Das Archivsystem wird von einem externen Dienstleister zur Verfügung gestellt, und wir haben noch keine Erfahrungen in der Zusammenarbeit, insbesondere wie schnell er auf Fragen reagiert.

Je besser du darin wirst, konkrete Fragen zu formulieren, desto leichter tust du dich im nächsten Schritt darin, konkrete Ziele für dein Vorgehen zu formulieren.

3.2.2 Was gilt es auszuprobieren? Was muss dein Produkt können?

Um deine Sorgen hinter dir und die Zuversicht in euer Produkt wachsen zu lassen, versuchst du, die zugrunde liegenden Fragen systematisch zu bearbeiten. Welche Teile eures Produkts benötigt ihr, um Antworten zu finden und auszuprobieren, was funktioniert und was nicht? Agile Entwicklung bedeutet so früh und so schnell wie möglich in und mit der Praxis zu lernen.

Hierfür ...	Brauchen wir das ...
Wir wissen noch nicht genau, in welchem Format und mit welcher Antwortzeit wir auf die Dokumente aus dem Archivsystem zugreifen können. Das hat unter Umständen Auswirkungen auf die Ausgestaltung des Anwendungsfall.	Der zeitkritischste Zugriff auf das Archivsystem erfolgt während der Kontoklärung telefonisch. Wir werden daher zu diesem Meilenstein den entsprechenden Anwendungsfall implementieren, um die Schnittstelle anzubinden und damit zu experimentieren, ob die Wartezeiten innerhalb des Ablaufs akzeptabel sind.

3.2.3 Darf es eine Nummer kleiner sein?

Je weniger Erfahrung du und dein Team mit agiler Entwicklung habt, desto größer ist die Wahrscheinlichkeit, dass ihr den Ausschnitt des Produkts im vorherigen Schritt noch größer als nötig gewählt habt.

Was könntet ihr im ersten Schritt noch weglassen, wenn es nur darum geht, mehr über das Risiko zu lernen, und nicht, dass das Umgesetzte komplett und schön ist? Welche Teile des Anwendungsfalls müsst ihr tatsächlich implementieren? Welche Teile der Schnittstellen müsst ihr umsetzen, welche könnt ihr noch weglassen?

3.2.4 Bring alles in einem Meilensteinplan zusammen

Formuliere nun jeweils einen Meilenstein für die von dir ausgewählten Punkte. Ein guter agiler Meilenstein erzählt, was du rund um eine kritische Frage im Projekt zu tun planst. Ein guter *Meilensteinplan* erzählt wiederum, wohin du dein Produkt als Erstes, Zweites, Drittes entwickeln möchtest, um Antworten auf die schwierigen Fragen zu bekommen. Dein Meilensteinplan ist eines der wichtigsten Instrumente, um allen Beteiligten ein Bild von den Risiken und Schwierigkeiten der Entwicklung zu geben, ohne dabei Angst und Schrecken zu verbreiten.

Wiederhole dieses Vorgehen noch ein paar Mal, bis du zumindest für alle großen Risiken ein Meilenstein definiert hast. Wenn du die Meilensteine jetzt in eine Reihenfolge bringst, sollten die größten Risiken möglichst am Anfang auftauchen (das klappt nicht immer zu 100 %, aber du solltest es versuchen). Wenn du sehr viele Meilensteine definiert hast, kann es auch Sinn ergeben, für die Planung einige der Meilensteine zusammenzulegen und so den Plan wieder etwas übersichtlicher zu gestalten.

Wenn du beginnst, Meilensteine zu suchen und in eine Reihenfolge zu bringen, ist es hilfreich, auch die in den folgenden Abschnitten beschriebenen Konzepte zu kennen, die sich in jedem Meilensteinplan wiederfinden lassen sollten.

3.2.5 Das Walking Skeleton

Der erste Meilenstein einer agilen Produktentwicklung sollte im Normalfall ein *Walking Skeleton* (dt. wanderndes Skelett) sein. Das Walking Skeleton besteht, wie der Name schon sagt, fast nur aus Knochen, also der Grundstruktur deines späteren Produkts, lose verbunden, sodass man es gerade so bewegen kann, ohne dass alles auseinanderfällt. Alistair Cockburn (Cockburn, 2008), einer der Urheber des Agilen Manifests, definiert es so:

> *A Walking Skeleton is a tiny implementation of the system that performs a small end-to-end function. It need not use the final architecture, but it should link together*

the main architectural components. The architecture and the functionality can then evolve in parallel.

(übersetzt*: Ein Walking Skeleton ist eine winzige Implementierung des Systems, die eine kleine End-to-End-Funktion ausführt. Es muss nicht die endgültige Architektur verwenden, aber es sollte die wichtigsten Architekturkomponenten miteinander verbinden. Die Architektur und die Funktionalität können dann parallel weiterentwickelt werden*.)

Die Funktion des Walking Skeletons ist nicht, dass es fachlich schon ernsthaft irgendwas könnte – es ist lediglich das technische Gerüst, in das im Weiteren die Features deines Produkts schrittweise eingefügt werden. Das Walking Skeleton sollte dennoch bereits potenziell lieferfähig sein. Das bedeutet, dein Team muss dafür Sorge tragen, dass euer Produkt bereits in diesem Stadium durch alle (automatischen) Qualitätssicherungs- und Bereitstellungsschritte von der Entwicklung bis zur Installation bewegt wird. Das ist initial eine Menge Arbeit, aber es stellt sicher, dass ihr Risiken und Probleme auf dieser Strecke frühzeitig erkennt und gleich zum Start damit beginnt, diese Prozesse robust zu gestalten und zu automatisieren. Agile Entwicklung ist auch die Fähigkeit, mit hoher Geschwindigkeit und Qualität auf Probleme zu reagieren. Das Walking Skeleton bildet dafür die Grundlage.

3.2.6 Das Minimum Viable Product (MVP) und das Minimal Marketable Product (MMP)

Zur agilen Produktentwicklung gehört die Idee eines Minimum-Produkts zentral dazu. Eric Ries (2009) hat dazu initial das Konzept des *Minimum Viable Products* (*MVP*) aufgebracht.

Das MVP ist eine Ausbaustufe eures Produkts, das dir und deinem Team ermöglicht, so viel praktisches Feedback von eurer Kundschaft zu bekommen wie möglich. Praktisches Feedback bedeutet dabei, dass eure Kundschaft in der Lage ist, das Produkt tatsächlich auf irgendeine Art und Weise schon zu benutzen. Auf »irgendeine Art und Weise« bedeutet nicht, dass euer Produkt schön aussehen muss, dass alle Funktionalitäten schon da sind oder dass ihr nicht im Hintergrund die Daten per Hand von einem System ins andere kopieren dürft. Es bedeutet nur, dass eure Kundschaft eine erste Idee davon bekommen kann, was euer Produkt tatsächlich auszeichnet.

Der Punkt des praktischen Ausprobierens ist deshalb so wichtig, weil wir uns als Menschen Neues nur schwer vorstellen können. Der typische Satz in dieser Phase ist immer: »Das ist doch wie (... hier irgendwas Altes, Bekanntes ...).« Erst beim Ausprobieren wird

uns klar, was tatsächlich neu ist, und wir fangen an, die Konsequenzen und Möglichkeiten zu verstehen.

Das MVP ist kein Ausbaustand, der wirklich fertig ist und den jemand wirklich haben möchte, aber es ist ein Stand, der euch in die Lage versetzen soll, die Idee eures Produkts zum ersten Mal für alle Beteiligten tatsächlich begreifbar zu machen, und das im Wortsinn.

Der nächste minimale Ausbaustand ist das *Minimal Marketable Product* (*MMP*). Das MMP ist der minimale Ausbaustand eures Produkts, das die zentralen Bedürfnisse eurer Kundschaft erstmalig befriedigt und ihr Problem löst. Dem MMP werden viele Komfortfunktionen noch fehlen, unter Umständen sogar noch ein paar Basisfunktionen, aber wenn die Zeit knapp wird, sollte es in der Lage sein, bei eurer Kundschaft so lange zu funktionieren, bis ihr die fehlenden Teile nachliefern könnt. Abbildung 3.3 zeigt, wie diese zusammenhängen.

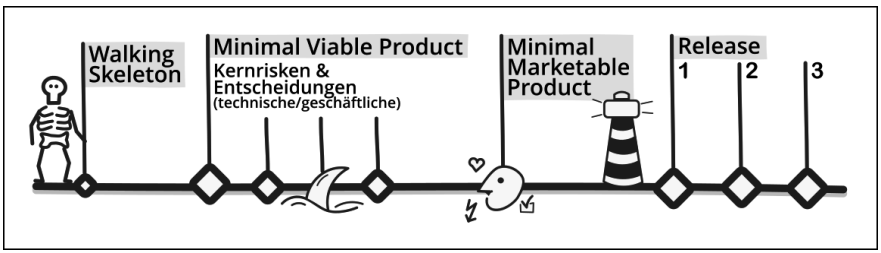

Abbildung 3.3 Die ersten Ausbaustufen eines Produkts bis hin zu den ersten Releases

Überlege dir als Teil deiner Vorgehensstrategie, wie für dein Produkt ein MVP und das MMP aussehen könnten. Unter Umständen ist es sinnvoll, mehrere MVPs für unterschiedliche Aspekte deines Produkts zu entwickeln.

Aus unserem Erfahrungsschatz: »Irgendwas Minimales«

Die Begriffe MVP und MMP stiften immer wieder viel Verwirrung. Insbesondere in Stakeholder-Ohren bleibt häufig nur ein »minimal« hängen bei gleichzeitig geringem Verständnis dafür, welche Ziele damit verbunden werden. Daher empfehlen wir, die Begriffe in der Kommunikation von Plänen nicht zu benutzen, sondern immer nur über die konkreten Inhalte und Ziele der mit dem MVP/MMP verbundenen Meilensteine zu sprechen.

Hinzu kommt, dass der Begriff MVP mittlerweile häufig synonym mit MMP benutzt wird. Daher merk dir vor allem die Ideen hinter den beiden Konzepten und versuche, für dein Produkt lieber eigene sprechende Namen für die jeweiligen Ausbaustufen zu finden.

3.2.7 Erzähle deine Vorgehensstrategie

Sobald du eine Reihe von Meilensteinen für dein Produkt gefunden und in eine sinnvoll erscheinende Reihenfolge gebracht hast, solltest du einmal testweise probieren, entlang der Meilensteine zu erzählen, wie du planst, bei der Entwicklung vorzugehen. Fang vorne an, was ist der erste Meilenstein, weshalb ergibt er an dieser Stelle Sinn? Was werdet ihr dann gelernt, abgesichert, bereitgestellt haben? Wie geht es dann weiter?

Versetze dich dazu in deine Stakeholder*innen und frage dich: Welche Argumente müssen sie hören? Hast du an alle gedacht?

3.3 Wie sag ich es den anderen?

Die Bedürfnisse der künftigen Nutzerschaft deines Produkts sind ausschlaggebend dafür, welche Features und Eigenschaften euer Produkt haben wird. Mindestens genauso wichtig sind deine Auftraggebenden und/oder das Management in deinem Umfeld. Sie entscheiden darüber, ob das Produkt überhaupt entwickelt wird, welche Ressourcen dir zur Verfügung stehen, und sie können dir unter Umständen dabei helfen, Hindernisse aus dem Weg zu räumen.

Ein vorausschauender, aussagekräftiger Meilensteinplan hilft dir, sie in deine Überlegungen einzubeziehen, sie mit den Risiken eures Vorhabens vertraut zu machen und aufzuzeigen, wie ihr plant, mit ihnen umzugehen. Was aus deiner Sicht klar und logisch erscheint, muss bei deinem Gegenüber noch lange nicht so ankommen. Deine Stakeholder*innen werden Menschen unterschiedlicher Expertise sein – von Technik und Betrieb über Recht und Finanzen, Marketing und Vertrieb bis Unternehmensführung und Strategie wird im Zweifel alles mit dabei sein. Alle bringen ihre eigenen Expertisen und Interessen mit ein, und zu deiner Product-Owner-Rolle gehört es, ihnen einerseits ein offenes Ohr zu schenken, ihre Ideen und Anforderungen mit aufzunehmen und andererseits deine Entscheidungen und Strategien klar zu kommunizieren.

In deiner Product-Owner-Rolle solltest du über die Strategie und Ausgestaltung deines Produkts allein entscheiden dürfen. Dazu brauchst du zwei Sachen:

- Das Vertrauen deines Umfelds, dass du ihre Interessen verstehst und berücksichtigst, sodass sie dir tatsächlich erlauben, Entscheidungen zu treffen, die sie mit betreffen.
- Die Fähigkeit, deine Überlegungen und Argumente passend darzulegen und Meinungsverschiedenheiten und Konflikte konstruktiv zu lösen.

Damit bist du nicht allein. In deinem Team hast du Menschen, die dir helfen, und auch dein*e Scrum Master*in wird dir zu Seite stehen. Sie oder er ist insbesondere auch dafür

verantwortlich, eurem Umfeld zu helfen, die agile Sicht- und Arbeitsweise zu verstehen. In Kapitel 6, »Zuhören, verstehen, ansprechen: dein Kommunikationsjob«, erfährst du noch mehr über deine Vermittlerrolle zwischen allen Beteiligten.

3.3.1 Die passende Argumentation

Deine Rolle als Product Owner*in bringt es mit sich, dass du immer wieder ganz unterschiedlichen Menschen nachvollziehbar erklären musst, was deine Entscheidungen motiviert, um sie mit an Bord zu holen. Die Entscheidungen, die du vertrittst, sind dabei normalerweise komplexe Entscheidungen, die den gesamten Kontext eures Vorhabens und die grundlegende Ausrichtung betreffen.

In Kontakt mit deinen Stakeholder*innen ist es daher wichtig, ihre jeweilige Perspektive einzunehmen und deine Argumentation auf die jeweilige Person abzustimmen. Eine unpassende Argumentation würde zum Beispiel so lauten:

Product Owner*in: *»Dieses Feature ist wichtig, damit sich das Team auf den neusten Stand der Technik bringen kann. Sie lernen dabei ein neues Framework, das interessant ist.«*

und Reaktionen wie diese hervorbringen:

Team: *»Cool!«*

CFO: *»Dafür kann ich den Preis des Produkts nicht erhöhen. Soll HR doch Leute anstellen, die schon ausgebildet sind.«*

CMO: *»Wie soll ich das denn in den Marketing-Flyer reinkriegen?«*

CEO: *»Nein.«*

Indem du in der Logik der Zielperson argumentierst, wird diese dir und deinen Argumenten besser folgen können. Du veränderst dabei nicht die Sache an sich, sondern du hebst bestimmte Aspekte daran hervor, also beispielsweise finanzielle Vorteile für CFOs oder Wettbewerbsvorteile für Marketing-Vertreter*innen, beispielsweise so

zum Team: *»Dieses Feature können wir mit dem neuen Framework umsetzen, das uns auch später einiges erleichtern sollte, auch wenn ihr euch jetzt erst mal einarbeiten müsst.«*

zum CFO: *»Dieses Feature erfordert die Weiterbildung unseres Teams. Das kostet X €, aber das rechnet sich schnell wieder. Wenn wir es nicht machen, verlieren wir den Anschluss, wenn es um State-of-the-Art-Entwicklung geht. Unsere Produkte werden sich schnell nicht mehr verkaufen lassen.«*

zum CMO: *»Mit der Arbeit an diesem Feature gewinnen wir die Fähigkeit, das Produkt so zu entwickeln, wie es unsere direkten Konkurrenten noch nicht können.«*

zum CEO: *»Mit diesem Feature legen wir die Basis für unsere künftige Vorreiterrolle.«*

Welche Art von Argumenten zu welchen Menschen passen, findest du im Verlauf der Arbeit mit ihnen heraus. Achte im Austausch darauf, auf welche Typen von Argumenten wer gut reagiert, und auch darauf, welche Argumente sie selbst für ihre Anliegen einsetzen. Gebräuchliche Typen sind (Frischherz et al., 2022):

- *Kausalargument* – Aufzeigen von Ursache und Wirkung
- *Induktionsargument* – Beispiele für gewünschte oder unerwünschte Eigenschaften
- *Analogieargument* – Vergleich mit einer ähnlichen Sache
- *Autoritätsargument* – Stützung durch eine Person/Institution, die als Autorität wahrgenommen wird
- *Teil-Ganzes-Argument* – Übertragung von gewünschten/unerwünschten Eigenschaften eines Teils auf die ganze Sache

Was für die Auswahl von Argumentationen gilt, gilt auch für die Auswahl von Beispielen: Bediene dich an Bildern aus dem Kontext deiner Zielperson oder aus dem allgemeinen Alltag, sodass sich alle mit den Beispielen wirklich identifizieren können. Dazu kann es auch hilfreich sein, beispielsweise Hobbys von anderen Personen zu kennen. Möglicherweise lässt sich daraus mal ein gutes Beispiel ableiten.

Ziel solcher Argumentationen ist es nicht nur, dass du deine Stakeholder*innen überzeugst, sondern auch, dass ihr überhaupt erst ins Gespräch kommt und euch offen über das Produkt austauschen könnt.

3.4 Vom ersten Meilenstein zum Product Backlog

Du hast dir jetzt erfolgreich eine Vorgehensstrategie für die Entwicklung deines Produkts erarbeitet. Wenn du im Laufe der Zeit mehr über deine Kundschaft und dein Produkt lernst, wirst du sie anpassen wollen, aber deine strategische Planung wird voraussichtlich recht lange stabil bleiben können. Jetzt ist es also Zeit, dir Gedanken zu machen, worum sich das Team in der nächsten Zeit kümmern soll. In Scrum bildest du das mithilfe des Product Backlogs ab.

3.4.1 Das Product Backlog

Das *Product Backlog* ist eine sortierte Liste der Dinge, die an deinem Produkt in nächster Zeit getan und verbessert werden sollen, damit es nützlich(er) und wertvoll(er) wird. Das Team leitet aus dem Product Backlog ab, was es im Sprint tut.

Um das Product Backlog zu befüllen, schaust du dir den nächsten anstehenden Meilenstein an und fragst dich, wie du die Dinge, die dafür zu tun sind, so runterbrechen kannst, dass sie innerhalb eines Sprints bearbeitet werden können, und in welcher Reihenfolge sie am besten getan werden sollten. Der Prozess des Runterbrechens, Beschreibens und Sortierens der Product-Backlog-Einträge ist das *Refinement*. Als eine deiner zentralen Verantwortungen haben wir dem Refinement ein ganzes eigenes Kapitel gewidmet (siehe dazu auch Kapitel 7, »Frisch sortiert ist halb gewonnen: das Refinement«).

Mit dem Product Backlog als Verfeinerung deiner strategischen Planung ergeben sich drei Planungsebenen.

3.4.2 Die drei Planungsebenen

Es ist hilfreich, dabei die folgenden drei Planungsebenen zu unterscheiden (siehe Abbildung 3.4):

- Die *strategische Planungsebene* – auf ihr liegen deine Produktvision sowie dein Meilensteinplan als Ausdruck deiner Vorgehensstrategie. Die Ziele auf dieser Ebene sind eher groß, grob und längerfristig.
- Die *taktische Planungsebene* – der Planungshorizont auf dieser Ebene umfasst ein paar (etwa drei bis sechs) Sprints auf dem Weg zum nächsten Meilenstein. Dein Product Backlog sollte mit genügend Einträgen dafür gefüllt sein
- Die *operative Planungsebene* – auf dieser Ebene plant das Entwicklungsteam, welche Product-Backlog-Einträge es im nächsten Sprint bearbeitet und wie es dabei vorgehen will. Der Planungshorizont ist beschränkt auf den jeweils anstehenden Sprint.

Du bist für die ersten beiden Planungsebenen verantwortlich. Mit dem Sprint Planning zum Beginn des Sprints geht die Verantwortung auf das Entwicklungsteam über. Das Team entscheidet, wie viel es meint, aus dem Product Backlog im Sprint schaffen zu können, und plant, was dafür zu tun ist. Die für den Sprint ausgewählten Product-Backlog-Einträge sowie der Plan zur Umsetzung (typischerweise einfach eine Sammlung von dafür zu erledigenden Aufgaben) werden als *Sprint Backlog* bezeichnet.

Gemeinsam formuliert ihr noch ein Sprint-Ziel für das Sprint Backlog.

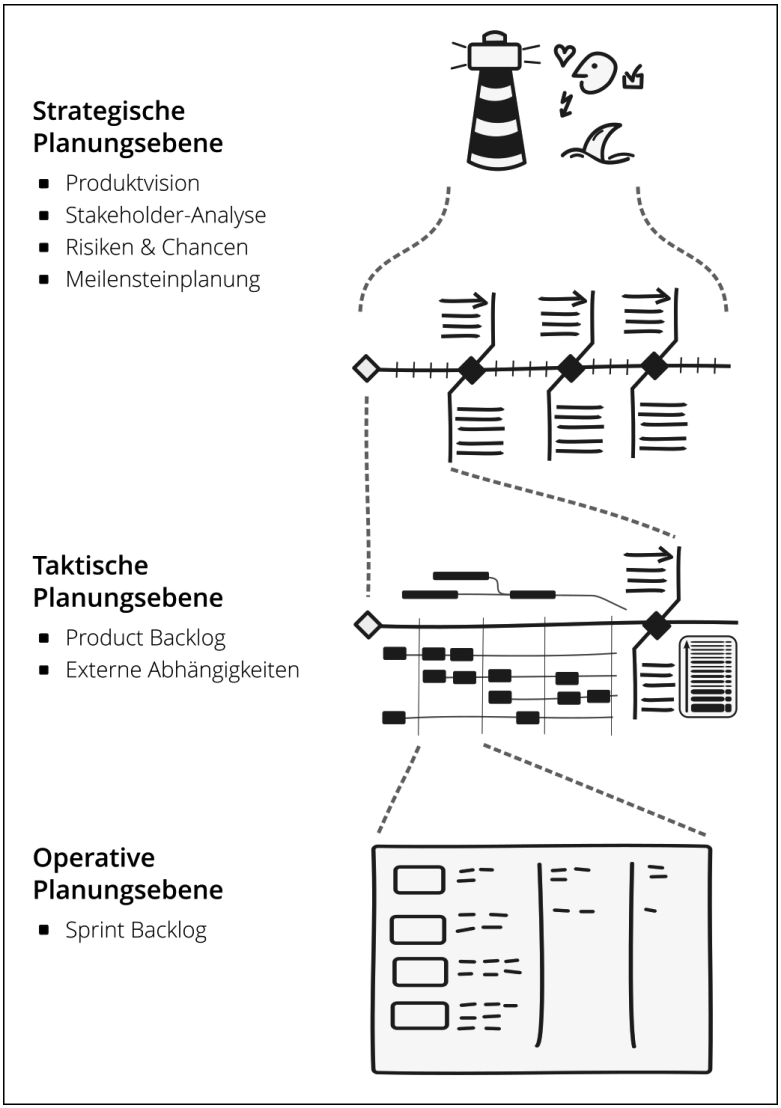

Abbildung 3.4 Die Planungsebenen auf dem Weg zum Product beziehungsweise Sprint Backlog

3.4.3 Sprint-Ziele

Sprint-Ziele verfolgen für den Sprint dasselbe Ziel im Kleinen wie die Meilensteine für dein Vorhaben im Großen: Was hofft ihr mit den ausgewählten Product-Backlog-Einträgen am Ende des Sprints erreicht zu haben?

Hast du dich im Rahmen der strategischen Planung gefragt, welche Fragen und Risiken bei der Entwicklung deines Produkts als Ganzes lauern, fragt ihr euch nun, welche Fragen und Risiken bei der Entwicklung eines ausgewählten Features lauern, beziehungsweise was möglich wird, wenn ihr diesen Entwicklungsstand erreicht.

Sprint-Ziele könnten also beispielsweise lauten: Am Ende des Sprints ...

- ... können wir die Navigationslogik in der Eingabemaske testen.
- ... können wir mit dem Betrieb die Konfiguration der Firewall beginnen.
- ... sollte der Ausbaustand gut genug sein, um ein Release für den Messestand bereitstellen zu können.
- ... sollte der Kernprozess gut genug abgebildet sein, dass wir uns ab dann auf die Sonderfälle konzentrieren können.
- ... soll es irgendwie laufen, hübsch kommt später!
- ... sollen Feature X und Y wirklich funkeln, wir wollen damit ins Fernsehen!

Das Entwicklungsteam fokussiert seine Arbeit während des Sprints in erster Linie auf das Sprint-Ziel. Stellt es fest, dass die Arbeit sich anders entwickelt als gedacht, wird es mit dir darüber sprechen, welche Kompromisse ihr machen könnt im Umfang oder der Art der Realisierung, ohne das Sprint-Ziel zu gefährden (siehe dazu auch Abschnitt 9.2, »Das Sprint-Ziel formulieren«).

Tipp: Sprint-Ziele vordenken

Der Scrum Guide sieht vor, dass das Sprint-Ziel erst während des Sprint Plannings gemeinsam mit dem Entwicklungsteam formuliert wird. Aus unserer Erfahrung heraus macht es allerdings Sinn, wenn du dir als Teil der taktischen Planung schon vorab Gedanken machst, wie die Sprint-Ziele grob aussehen können.

Das hilft dir beim Refinement des Product Backlogs und vermittelt dem Team gleich zu Beginn des Sprint Plannings eine Idee, weshalb das Backlog aktuell so aussieht, wie es aussieht. Ihr könnt und solltet dann während des Sprint Plannings das Sprint-Ziel immer noch gemeinsam anpassen und prüfen, ob das Team denkt, es realistischerweise erreichen zu können.

3.5 Produktarten und ihre Herausforderungen

Unterschiedliche Produktarten bringen unterschiedliche Herausforderung in ihrer Entwicklung mit, auf die du dich am besten von Anfang an einstellen solltest. Du findest an

dieser Stelle einen Überblick über eine Reihe typischer Produktarten, verbunden mit beispielhaften Tipps und Empfehlungen zum Vorgehen in der Entwicklung. Nimm unsere Hinweise als einen Startpunkt. Uns ist klar, dass man zu jeder Produktart noch viel mehr sagen könnte. Konkretisiere und ergänze deine Strategie daher auf Basis deiner Erkenntnisse aus der Übersichtsphase und in enger Zusammenarbeit mit deinem Team.

Schaue dich am besten auch bei den Produktarten um, in die du dein Produkt nicht als Erstes einordnen würdest. Du findest sicher auch dort noch Aspekte, die relevant für dein Produkt sind.

3.5.1 (Web-)Apps

Apps sind Anwendungen, die auf Smartphones oder Tablets ausgeführt werden und idealerweise dabei selbsterklärend und sehr einfach zu bedienen sind. Im Unternehmenskontext sollen Apps typischerweise Kund*innen ermöglichen, wichtige Geschäftsprozesse selbst auszuführen, um einerseits Anliegen schneller zu erledigen und andererseits natürlich auch Verwaltungskosten zu sparen. Die Nutzung von Apps erfolgt (auch) mobil, bei eingeschränkter Netzwerkverfügbarkeit und unruhiger Umgebung wie etwa in der U-Bahn. Dabei laufen Apps in der Regel auf privaten Geräten, was die Fehleranalyse erschwert. Zu guter Letzt entwickeln sich die Entwicklungswerkzeuge und Softwarebibliotheken im App-Umfeld sehr dynamisch und stetig weiter, und die Hersteller erzwingen regelmäßige Aktualisierungen.

Strategieempfehlung

Die folgenden Punkte solltest du mit deinem Team möglichst früh realisieren, um eine solide Basis für eure Entwicklung zu schaffen:

- Beginne so früh wie möglich damit, eure Anwendung von fremden Nutzenden ausprobieren zu lassen. Das ist der einzige Weg, um herauszufinden, ob sie Interesse haben, eure App überhaupt zu nutzen, und ob sie verstehen, wie sie funktioniert.
- Sorge dafür, dass ihr den Qualitätssicherungs- und Lieferprozess von Anfang an automatisiert und optimiert. Ihr müsst in der Lage sein, häufig und sehr schnell liefern zu können, um mit Features experimentieren und Probleme beseitigen zu können.
- Erarbeitet frühzeitig Wege zum Nutzungs- und Fehler-Logging. Eure Lösungen müssen datenschutzkonform, performant und nützlich sein. Das ist kniffliger, als es zunächst scheint, und erfordert auch einiges an Infrastruktur.

3.5.2 Desktop-Anwendungen

Desktop-Anwendungen, Programme, die auf einem Arbeitsplatzrechner installiert werden, haben typischerweise einen größeren Funktionsumfang als Apps und richten sich damit an Expert*innen. Diesen kann man einerseits mehr zumuten in der Bedienung, das heißt, du kannst von einer Bereitschaft ausgehen, dass sie sich in die Arbeit mit eurer Anwendung einarbeiten, andererseits erwarten sie dafür auch durchdachte und optimierte Oberflächen, denn sie werden eure Anwendungen voraussichtlich den gesamten Arbeitstag über intensiv nutzen.

Expert*innen kennen ihre Arbeitsprozesse sehr viel besser, als du und dein Entwicklungsteam es je tun werden, und gleichzeitig sind sie damit meist auch nicht in Lage, alle ihre Anforderungen bewusst zu kommunizieren (»Ist doch klar, dass man das so braucht …«). Sofern ihr euch entscheidet, etablierte Arbeitsprozesse umzugestalten, musst du darauf achten, dass ihr auch den dafür notwendigen Change-Prozess mitgestaltet. – Ein Produkt funktioniert nur, wenn es auch die Nutzerschaft mitnimmt.

Strategieempfehlung

Für gute Desktop-Anwendungen gilt erst mal dasselbe wie für Apps – mit noch ein paar anderen Schwerpunkten:

- Beginne wieder so früh wie möglich damit, eure Anwendung durch Nutzer*innen ausprobieren zu lassen. Stellt euch darauf ein, dass ihr dabei viele Anforderungen an euer Produkt erst noch entdecken werdet, egal wie viel Prozessanalyse ihr vorher schon getrieben habt.
- »Komfortfunktionen« wie Autovervollständigungen, Tastaturkürzel und Drag-and-drop-Möglichkeiten haben in Anwendungen für Expert*innen den Charakter von Basisanforderungen, integriert diese stetig mit und nicht erst als separate Ausbaustufe.
- Wenn ihr existierende Anwendungen ablöst, achtet darauf, eure Nutzerschaft mitzunehmen. Schulungen, Hilfsmaterialien und frühzeitige Informationsangebote sollten eure Entwicklung begleiten, und du solltest sie als Teil deines Produkts und damit deiner Verantwortung begreifen. Erprobt und testet diese Dinge wie alle anderen Aspekte eures Produkts auch.
- Bei der Bereitstellung von Desktop-Anwendungen können eine Menge Probleme auftreten. Eventuell müsst ihr außerdem gewährleisten, dass ihr unterschiedliche Versionen noch parallel betreiben und betreuen könnt. Entwickelt und testet daher auch diese Verfahren wiederum so früh wie möglich.

3.5.3 Backend

Backend-Systeme verrichten im Hintergrund ihre Arbeit, verwalten und verarbeiten große Datenmengen, und das häufig über Jahrzehnte hinweg. Das bedeutet auch, dass ihr fast nie auf der grünen Wiese arbeitet – Altdaten müssen migriert, andere Systeme angeschlossen werden, und dein Entwicklungsteam wird häufiger vor der Frage stehen: »Ist das Kunst, oder kann das weg?«

Wenn im Backend etwas schiefgeht, die Performance einbricht oder Fehler passieren, sind die Auswirkungen oft weitreichend und der Schaden entsprechend groß.

Strategieempfehlung

Für Backend-Systeme gilt, dass du von Anfang an ein besonderes Augenmerk auf die Entwicklung der Betriebsprozesse legen solltest.

- Entwickle Backup-, Restore- und Migrationsroutinen so früh wie möglich, genauso wie Monitoring und Überwachungsprozesse für den Betrieb. Sie beeinflussen einerseits die Architektur eures Systems stark, andererseits helfen sie euch auch bei der Analyse von Problemen in der Entwicklung.
- Macht euch direkt zum Start Gedanken über Testumgebungen, Management und Konfiguration der Umgebungen und wie ihr die dazugehörigen Testdaten generieren und managen könnt. Ihr werdet auch Umgebungen für Last- und Performancetests benötigen! Die Investitionen in diesen Bereich mögen kurzfristig hoch erscheinen, aber sie zahlen sich mittel- und langfristig immer wieder aus.
- Auf Backend-Systemen laufen in der Regel unternehmenskritische Prozesse über Jahre hinweg, eine gut aufgesetzte und gepflegte Testautomatisierung ist essenziell für die langfristige Wartung und um bei Problemen schnell reagieren zu können.
- Datensicherheit und Datenschutz sind Themen, die sehr viel leichter fallen, wenn ihr sie von Anfang an mitdenkt und realisiert.

3.5.4 IT-Modernisierung

Für viele Unternehmen ist die Modernisierung ihrer IT-Anwendungs- und Systemstruktur ein wichtiges strategisches Ziel, um die effektive Realisierung neuer digitaler Geschäftsmodelle und Dienste zu ermöglichen. Backend-Systeme müssen geöffnet werden, um Apps und Web-Anwendungen den Zugriff zu ermöglichen. Dabei gibt es häufig nicht mehr genügend Mitarbeitende mit tiefgehendem Know-how der eingesetzten Technologien und Frameworks, um anstehende Projekte zu besetzen. Letztlich versprechen neue Plattformen und Lösungen auch neue Möglichkeiten Mehrwerte zu generieren.

Zu bereits beschriebenen Herausforderungen aus den Bereichen »Backend«, »Desktop« und »(Web-)App« kommen insbesondere zwei neue Aspekte hinzu:

- Es besteht ein hoher Druck, Modernisierungs- und Neuentwicklungsvorhaben miteinander zu verknüpfen, um die Investitionen zu rechtfertigen. Das erhöht die Komplexität und die Risiken akkumulieren sich.
- Ein zentrales Ziel der Modernisierung ist in der Regel auch, mehreren Teams gleichzeitig zu ermöglichen, Anforderungen in den zentralen Serversystemen umzusetzen. Das erfordert regelmäßig neue und robustere Entwicklungs- und Qualitätssicherungsprozesse.

Strategieempfehlung

IT-Modernisierungsvorhaben sind komplexe Großprojekte mit vielen Stakeholdern und weitreichenden Auswirkungen ins Unternehmen hinein. Neben den zu den anderen Produktarten bereits gegebenen Empfehlungen, solltest du darauf achten:

- Versuche neue Features in der Breite erst nach einer Erneuerung der betroffenen Anwendungen und Systemen umzusetzen, um nicht zu viele Baustellen gleichzeitig aufzumachen.
- Entwickle eine Roadmap die aufzeigt, welche Risiken ihr durch diese Vorgehensweise mitigiert.
- Führe Mitarbeitende aller betroffenen Anwendungen und Systeme, alt wie neu, zu einem echten agilen Team bei dir zusammen. Ihr braucht den engen Austausch, um neue Lösungswege schnell erarbeiten und verifizieren zu können.
- Binde Stakeholder*innen aus der Organisation frühzeitig mit in die Reviews ein. Eure Arbeiten werden Auswirkungen auf viele interne Arbeitsprozesse haben und ein direkter, offener Kommunikationsfluss beugt vielen Problemen und Missverständnissen vor.
- Modernisierungsvorhaben sind immer auch Change-Projekte. Hole dir nach Möglichkeit entsprechende Unterstützung mit ins Team und mache dir bewusst, dass diese Aufgabe mit zu deinem Produkt gehört.

3.5.5 Embedded-/Steuerungssysteme

Embedded- oder Steuerungssysteme verrichten ihre Arbeit im Inneren von Geräten und Anlagen. Sie agieren dabei weitgehend autonom auf der Basis von Sensordaten, die sie häufig in Echtzeit verarbeiten müssen.

Während entsprechende Systeme auf der einen Seite sehr robust und verlässlich sein müssen, sind sie auf der anderen Seite mit einer realen Welt konfrontiert, die komplex, vielfältig und selten in allen Facetten vorhersehbar ist. Dabei müssen sie auch in unerwarteten Situationen stets sinnvoll reagieren.

Strategieempfehlung

- Für Embedded- und Steuerungssysteme gilt einmal mehr, dass ihr so früh wie irgend möglich damit beginnen solltet, unter realen Bedingungen zu testen, also mit echter Hardware unter möglichst vielfältigen Umweltbedingungen. Die Realität ist euer größter Lehrmeister! Legt früh den Fokus darauf, wie ihr (Sensor-)Daten aus dem Betrieb aufzeichnen und für die Fehleranalyse und Tests wieder abspielen könnt. Ihr legt damit die Basis für reduzierbare und robuste automatisierte Tests.
- Es gibt für die Entwicklung von Steuerungssystemen auch sehr gute und bewährte formelle Analyse- und Designmethoden, mit denen dein Team vertraut sein sollte. Agile Produktentwicklung heißt nicht, einfach nur wild rumzuprobieren, sondern eure Analysen und Designs so früh und so oft wie möglich praktisch zu überprüfen, um Lücken und Fehlannahmen aufzuspüren. Nehmt euch Zeit für die entsprechenden Methoden und verlängert dafür im Zweifel lieber euren Sprint-Zyklus.
- Wenn ihr Steuerungstechnik oder Embedded-Systeme entwickelt, seid ihr mit großer Wahrscheinlichkeit auch in einem regulierten Umfeld unterwegs. Beginnt gleich zu Beginn damit, euch mit den entsprechenden Anforderungen an Dokumentation und Qualitätssicherung vertraut zu machen, damit ihr diese in eurem Entwicklungsprozess mit verankert und nicht am Ende noch ein Haufen Arbeit auf euch wartet.
- Auch vermeintlich kleine Änderungen an der Hardware eures Produkts können große Auswirkungen haben. Integriert daher alle eure Produktkomponenten so oft wie möglich und macht gemeinsame Reviews eurer Arbeit.

Wenn Planung und Vorgehensstrategie stehen, ist das ein guter Zeitpunkt für Feedback von außen. Wie du das gestalten kannst, steht im nächsten Kapitel.

Fokus! Das Handbuch für Product Owner

Du bist Product Owner*in in der IT. Deine Aufgabe könnte kaum komplexer sein, und vom Überblick zum Projektstart über gute Kommunikation bis zur Produktqualität hängt vieles von deiner Arbeit ab.

Mit diesem Handbuch füllst du die Rolle professionell aus. Du lernst, die Prinzipien und Methoden von Scrum zielführend einzusetzen. Für den Einstieg und als Begleiter in der Praxis.

+ Lebendige Theorie
+ Robuste Methoden für verschiedene Projektsituationen
+ Moderationstechniken mit vielen Beispielen
+ Handfeste Tipps
+ Projektleiter*innen, Agile Coaches und Product Owner*innen

Aus dem Inhalt

»Ein Buch wie ein gutes Vorbild. Als wäre jemand an deiner Seite, der deine Aufgaben gut kennt und zum ständigen Weiterlernen anspornt.«

Julia Dellnitz, **Jan Gentsch**, **Dr. Sascha Demarmels**, **Dina Sierralta** und **Uwe Vigenschow** sind erfahrene Projektleiter*innen, Agile Coaches und Product Owner*innen. Sie arbeiten für ein zielführendes Miteinander, wertschätzende Kommunikation und Technologie für Menschen. In diesem Buch teilen sie ihre langjährige Projekterfahrung mit euch.

Rheinwerk Computing

€ 34,90 [D] € 35,90 [A]

Gedruckt in Deutschland
Papier aus nachhaltiger Waldwirtschaft
Mineralölfreie Druckfarben

Informatik/Softwareentwicklung
ISBN 978-3-8362-9269-6