Agile Roadmaps

Agile Ansätze wie Scrum thematisieren die Entwicklung im Team, sagen aber nichts darüber aus, wie die Verbindung ins Unternehmen aussieht. Für die unumgängliche Integration wichtiger Stakeholder wie z. B. Sponsoren oder Lenkungsausschüsse scheint es naheliegend, die Instrumente zu verwenden, die sich auf Teamebene bewährt haben. Leider funktionieren Story Points, Velocity und Release Burnup Charts nicht mehr gut, wenn sie den Teamkontext verlassen.
Dieser Artikel erzählt die Geschichte eines Projektes, das Story Points, Velocity und Release Burnup Charts für die Interaktion mit dem Lenkungsausschuss nutzte und in Schieflage geriet. Durch andere, fachlich-inhaltliche Instrumente wie zielorientierte Roadmaps und Parking Lot Diagramme konnte es schließlich einen besseren Kontakt zu wichtigen Stakeholdern herstellen. Von Stefan Roock

Es war einmal ein agiles Projekt ...

Es war einmal ein hoffnungsfrohes Projekt, das ein cooles neues Produkt entwickeln wollte. Man bildete drei Scrum-Teams mit jeweils eigenem Product Owner, weil man schnell vorwärtskommen wollte.

Nun stellte sich die Frage, wie mit dem Projektleiter umzugehen sei, der gegenüber dem Unternehmen den Projekterfolg verantwortete. Offensichtlich war dieser oberhalb der Product Owner anzusiedeln. Er sollte natürlich nicht durch Micro-Management die Arbeit der Scrum-Teams stören, so dass er nur in engen Grenzen direkt mit dem Team interagieren durfte. Die Kommunikation sollte im Wesentlichen über die Product Owner laufen.

Der Projektleiter hielt den Kontakt zum Lenkungsausschuss, um diesen über den Projektfortschritt auf dem Laufenden zu halten. Abbildung 1 zeigt die gewählte Projektstruktur.

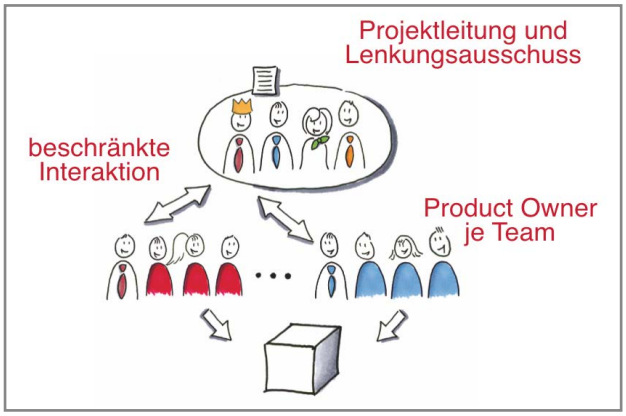

Abb. 1: Projektstruktur

... das wurde immer schneller ...

Natürlich setzte man im Projekt wie üblich Story Points zum Schätzen und Release Burnup Charts für die Fortschrittskontrolle ein. Und tatsächlich lief das Projekt gut. Die Teams wurden immer schneller und so konnte der Projektleiter bald im Lenkungsausschuss verkünden, dass man sogar früher als geplant fertig sein würde (Abb. 2).

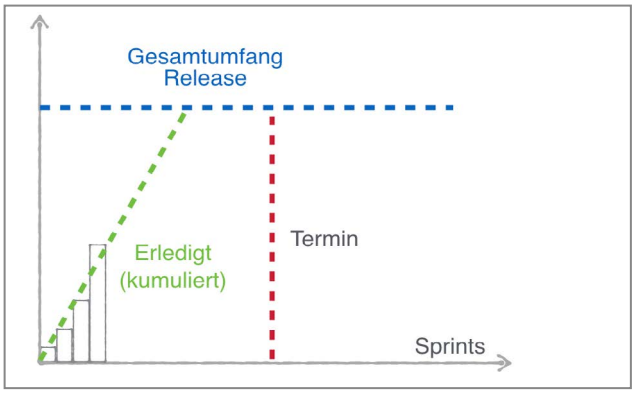

Abb. 2: Release Burnup (die Teams werden immer schneller)

aber dann...

Ein paar Sprints später stellte sich die Situation plötzlich ganz anders dar. Das Team war plötzlich viel langsamer geworden und gleichzeitig stieg der Gesamtumfang des Systems kontinuierlich an (Abb. 3). Der ursprünglich geplante Termin war nicht nur völlig unrealistisch geworden, sondern viel schlimmer noch: Ein neuer Termin konnte nicht prognostiziert werden. Woher soll man in dieser Situation auch abschätzen können, wie sich die Entwicklungsgeschwindigkeit des Teams und der Gesamtumfang des Systems weiterhin entwickeln würde? Würde sich die Situation irgendwann wieder normalisieren oder befand man sich auf einem Todesmarsch (Death March)?

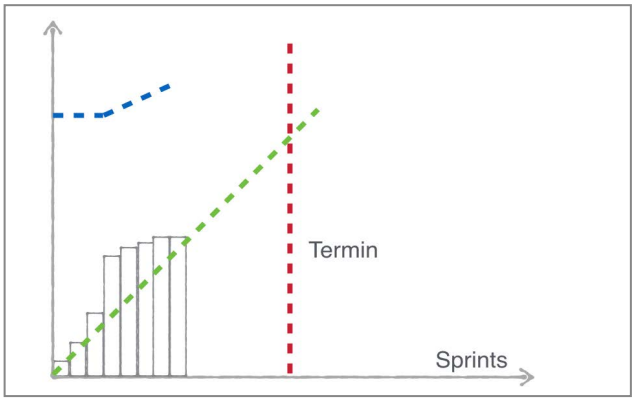

Abb. 3: Death March?

Story Points

Story Points sind ein Schätzansatz, der in der agilen Welt sehr verbreitet ist. User Stories werden mit Story Points relativ zueinander geschätzt. Eine Referenzstory bekommt 3 oder 5 Story Points. Für die anderen User Stories wird dann diskutiert, wie groß diese im Verhältnis zur Referenzstory sind. Mit Story Points wird also nur die Größe der User Stories geschätzt, aber nicht die Produktivität des Teams. Diese wird gemessen, indem für einige Sprints beobachtet wird, wie viele Story Points das Team erledigt (Velocity). Über die Kombination von Story Points und Velocity kann errechnet werden, wie lange die Entwicklung ungefähr dauern wird.

Release Burnup Chart

Das Release Burnup Chart zeigt den quantitativen Fortschritt bei der Entwicklung eines Releases und erlaubt eine grobe Abschätzung über den weiteren Verlauf. Ein Beispiel in Abbildung 2: die schwarzen Balken zeigen die kumulierten Story Points der bereits fertig entwickelten User Stories. Die gestrichelte grüne Linie zeigt den durchschnittlichen Fortschritt. Die gestrichelte blaue Linie zeigt den geplanten Gesamtumfang des Systems in Story Points. Dort, wo sich grüne und blaue Linie treffen, liegt ein plausibler Fertigstellungstermin. Die rote Linie zeigt den ursprünglichen anvisierten Termin.

Sicher war nur, dass man nicht einfach so weitermachen konnte. Also hielt man das Projekt an und analysierte die Ursachen.

Und die Moral von der Geschicht'

Bei der Analyse wurden die folgenden Punkte als problematisch identifiziert:

- Kunden aus den Augen verloren
- Story-Point-Wettrennen der Teams
- Technische Schulden
- Diagramm-Spekulationen

Kunden aus den Augen verloren

Im Fokus der Entwicklung stand das System mit seinen Features und nicht die Endkunden mit ihren Bedürfnissen. So war das Erfolgskriterium im Team nicht mehr Kundenzufriedenheit, sondern alle definierten Features „in time" zu liefern. Dadurch fehlte ein wichtiger Hebel für den Erfolg: Man konnte nicht mit Hinblick auf die Kundenbedürfnisse priorisieren und entscheiden, bestimmte Features nicht umzusetzen.

Story-Point-Wettrennen der Teams

Der „Zwang" alle Features zu liefern, kombiniert mit dem sportlichen Zeitplan des Projektes, erzeugte hohen Leistungsdruck und rückte die Entwicklungsgeschwindigkeit in den Fokus. Das wurde weiter dadurch befördert, dass eine Art Story-Point-Wettrennen zwischen den Teams veranstaltet wurde. Die Teams versuchten sich mit ihrer Velocity (Story Points pro Sprint) gegenseitig zu übertrumpfen.

Technische Schulden

Dieses Story-Point-Wettrennen hatte seinen Preis: Die Teams erreichten immer höhere Geschwindigkeiten, indem sie die technische Qualität der Software vernachlässigten. Irgendwann wurden Änderungen am System immer aufwendiger und brachten die Teams fast zum Stillstand.

Diagramm-Spekulationen

All das war im Lenkungsausschuss so nicht bekannt. Man betrachtete etwas ratlos das Release Burnup Chart und versuchte, aus dem Diagramm Rückschlüsse zu ziehen und Maßnahmen abzuleiten. Die naheliegende Schlussfolgerung war stets: „wir müssen schneller werden". Und das ist leider nicht auf kurzfristig möglich.
Es fehlte an echtem Dialog, Lernen und inhaltlichen Diskussionen.

Auf ein Neues ...

Auf Basis der Analyse wurde das Projekt neu aufgesetzt und sollte sich am agilen Kernzyklus orientieren (Abb. 4): Agile Teams lösen Probleme ihrer Endkunden (und setzen nicht einfach Anforderungen um). Das tun sie in möglichst kurzen Zyklen und optimieren sowohl Produkt wie auch Prozess selbständig.

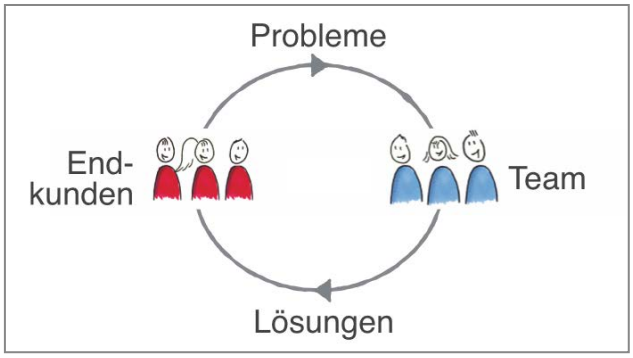

Abb. 4: Agiler Kernzyklus

In der Konsequenz wird das Scrum-Framework zum Hilfsmittel, um den agilen Kernzyklus zu etablieren: Der Kernzyklus bleibt im Vordergrund, das Scrum-Framework tritt in den Hintergrund (Abb. 5).
In vielen Projekten tritt leider das Scrum-Framework in den Vordergrund und verdrängt den Kundenfokus des ▶

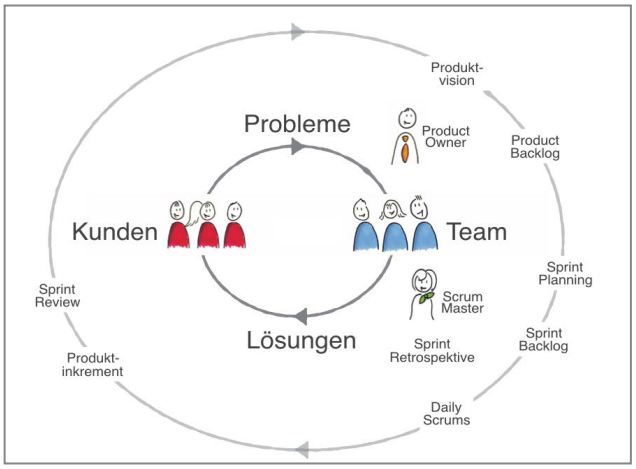

Abb. 5: Scrum-Framework im Hintergrund

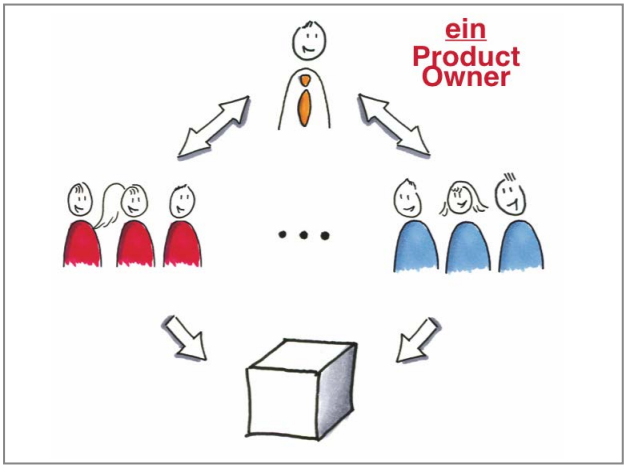

Abb. 6: Ein Produkt, ein Product Owner

agilen Kernzyklus. Dann vollführt man dogmatisch die Scrum-Rituale und vergisst, welchem Zweck die Scrum-Rollen, -Meetings und -Artefakte dienen. Im beschriebenen Projekt wurde an dem Verständnis gearbeitet, dass Scrum ein Mittel zum Zweck ist und auch so verwendet werden sollte.

Vereinfachte Struktur

Vor dem Hintergrund des agilen Kernzyklus wurde die Projektstruktur vereinfacht: Der Projektleiter wurde als Product Owner für alle drei Teams etabliert und die bisherigen Product Owner wurden zu Teammitgliedern (Abb. 6). Dadurch wurde der Abstand des verantwortlichen Projektleiters zu den Entwicklern deutlich reduziert.

Natürlich kann ein Product Owner in diesem Kontext nicht die User Stories für alle drei Teams schreiben und benötigt dazu die Unterstützung der Teams. Gemeinsame Workshops lieferten eine gute Basis und sorgten dafür, dass nicht nur der Product Owner, sondern auch die Teams die Bedürfnisse der Endkunden verstanden. Nur so konnten sie gemeinsam sinnvolle Lösungen konzipieren.

Los, zusammen jetzt

So begann der Neustart des Projektes mit mehreren Workshops zwischen Product Owner und den Teams, in denen Story Maps erstellt wurden. Die Workshops wurden im ständigen Wechsel zwischen Plenumsarbeit mit allen 25 Teilnehmern und Kleingruppenarbeit durchgeführt. Die Story Map füllte am Ende eine ca. acht Meter lange Wand in einem großen Besprechungsraum. Viel wichtiger als die erstellte Story Map war allerdings das gemeinsame Verständnis der Arbeitsweise der Endkunden und der anvisierten Lösung.

Wirksam werden

Nachdem das Big Picture mit Story Mapping erstellt war, haben die Beteiligten auf Basis der Story Map die Releases definiert. Die Releases wurden dabei ausgehend von angestrebten Wirkungen definiert. Schließlich wollte man mit der entwickelten Software eine Wirkung erzielen – für die Endkunden sollte sich etwas ändern.

Im Runtastic-Beispiel (im Kasten beschrieben) wären potenzielle Wirkungen:

- letzten Lauf auswerten
- Trainingsfortschritt auswerten

Story Maps

Story Maps nach Jeff Patton (siehe [Patton2014]) sind im Grunde eine spezielle Form von User Stories. Auch bei Story Maps geht es um das Erzählen einer Geschichte aus Benutzersicht. Die Geschichte wird entlang von User Tasks erzählt, die von links nach rechts angeordnet werden (siehe Abbildung 7). User Tasks sind die Aufgaben, vor denen der Benutzer steht (nicht zu verwechseln mit den technischen Tasks, die die Entwickler im Sprint Planning erstellen). Unterhalb der User Tasks werden Details in sogenannten Sub-Tasks festgehalten.

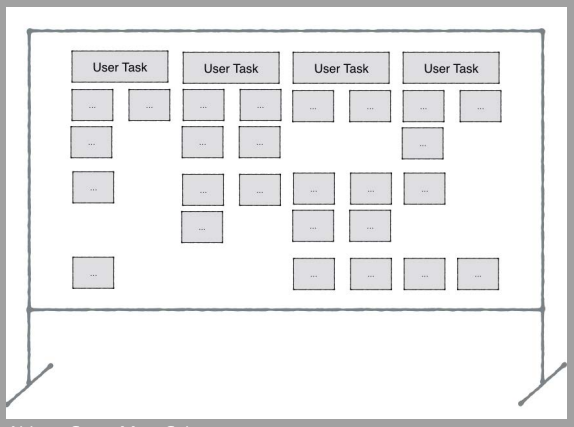

Abb. 7: Story Map-Schema

Abbildung 8 zeigt ein Beispiel für eine einfache Story Map. Die User Tasks sind rot markiert (Registrieren, Einloggen, Laufen, Auswerten, Planen). Die Sub Tasks darunter sind blau markiert.

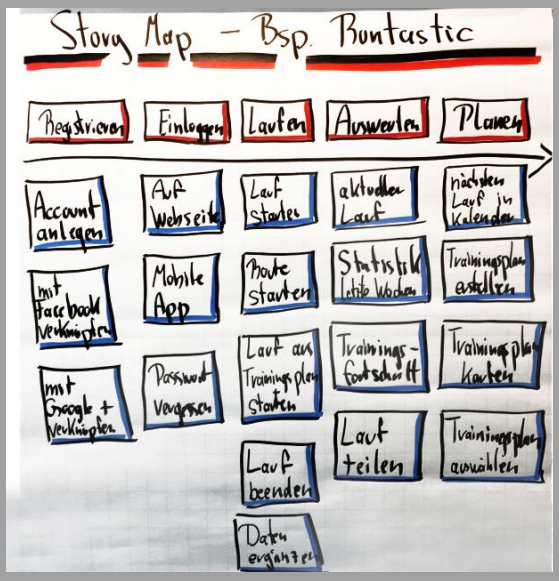

Abb. 8: Beispiel einer Story Map

- gezielter trainieren mit Trainingsplan

Die Wirkungen werden als Zeilen in der Story Map aufgeführt und die Sub-Tasks den jeweiligen Wirkungen zugeordnet (Abb. 9).

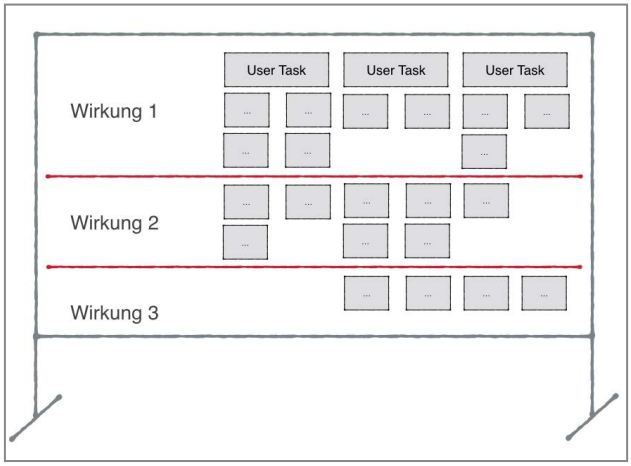

Abb. 9: Releases nach Wirkungen schneiden

Das Schneiden von Releases ausgehend von angestrebten Wirkungen funktioniert natürlich nicht nur mit Story Maps. Es funktioniert auch, wenn das Product Backlog mit Impact Mapping (siehe [Adzic2012]) erstellt wurde oder einfach aus Epics und User Stories besteht.

Und die Roadmap?

Wenn wir Releases anhand von Wirkungen definieren, haben wir bereits einen wichtigen Schritt in Richtung einer zielorientierten Roadmap unternommen (Abb. 10). Wir ergänzen je Wirkung die Metriken, an denen wir feststellen, dass die Wirkung eingetreten ist, den Termin sowie die Key-Features. Der Hauptnutzen der Metriken besteht in der Klärung dessen, was genau die Wirkung ist.

Reicht es aus, wenn ein Nutzer seinen Lauf auswerten kann oder sollen es tausende sein, die dafür bezahlen? Die Features sind die notwendigen Produkteigenschaften, um die Wirkung zu erzielen. Sie sollten so grobgranular und ▶

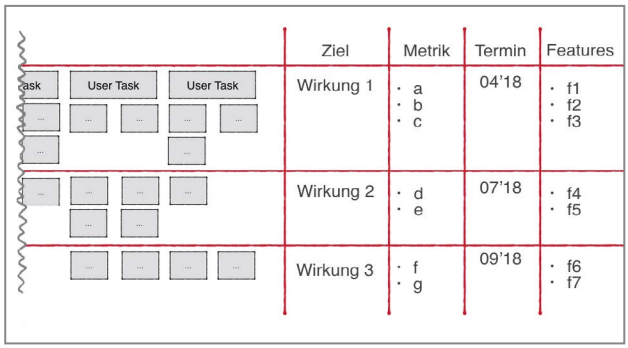

Abb. 10: Zielorientierte Roadmap

verständlich beschrieben sein, wie man sie auf einen Produktkarton drucken würde.

So haben wir aus der Story Map eine zielorientierte Roadmap abgeleitet, die dem Konzept der GO Roadmap von Roman Pichler sehr ähnlich ist (siehe [Pichler2016]).

Natürlich kommt man bei der Erstellung einer zielorientierten Roadmap nicht um eine Abschätzung herum. Anders könnte man die Terminspalte nicht sinnvoll befüllen. Diese Schätzung sollte allerdings so leichtgewichtig wie möglich erstellt werden.

Insbesondere sollte berücksichtigt werden, wie wahrscheinlich die späteren Releases in der Roadmap wirklich sind. Nehmen wir als Beispiel die Entwicklung eines neuen Produktes für einen Endkundenmarkt. Nach dem ersten Release sind zwei Extremszenarien denkbar: Im ersten Extremszenario interessiert sich kein Kunde für das Produkt. Dann muss das Team darüber reflektieren, ggfs. Kundenbedürfnisse besser verstehen und sein Vorgehen anpassen. Auf keinen Fall ergibt es Sinn, die Roadmap weiter umzusetzen. Im zweiten Extremszenario wird das Produkt am Markt ein Hit und in kurzer Zeit benutzen viele Kunden das Produkt. In diesem Fall wird das Team überschwemmt werden mit neuen Ideen der Kunden und Erkenntnissen aus dem Produkteinsatz im echten Leben. Auch in diesem Szenario wird man die Roadmap vermutlich nicht wie geplant umsetzen, sondern an die Gegebenheiten anpassen.

Zielorientierte vs. Feature-orientierte Roadmaps

In der Praxis sind Feature-orientierte Roadmaps die Regel. Die sehr große Menge aller gewünschten Produkteigenschaften wird aufgeteilt in mehrere große Mengen an Produkteigenschaften und Terminen zugeordnet. Man geht also von Produkteigenschaften (Features) aus.

Bei den hier vorgestellten zielorientierten Roadmaps gehen wir von Wirkungen bzw. businessrelevanten Zielen aus und beschreiben die zugehörigen Features nur soweit wie zum Verständnis beim Lenkungsausschuss oder Sponsor notwendig.

Tabelle 1 zeigt die Unterschiede zwischen feature- und zielorientierten Roadmaps.

	Feature-orientierte Roadmaps	Zielorientierte Roadmaps
Der Fokus liegt auf ...	... dem Produkt.	... dem Kunden und den eigenen Geschäftszielen.
Die Features sind ...	... feingranular.	... grobgranular.
Die Menge der Features ist ...	... groß.	... klein.
Erfolg bedeutet,	... die Features zum Termin umgesetzt zu haben.	... die Wirkung erzielt zu haben.
Änderungen am Produkt Backlog ...	... bedeuten meist eine Änderung der Roadmap.	... sind oft ohne Änderung der Roadmap möglich.

Tabelle 1: Feature- vs. zielorientierte Roadmaps

gewisser Weise ist eine Roadmap sachlich also unnötig. Sie ist häufig schlicht ein Zugeständnis an die Unternehmensrealität. Und so sollte sie gehandhabt werden: leichtgewichtig und leicht änderbar.

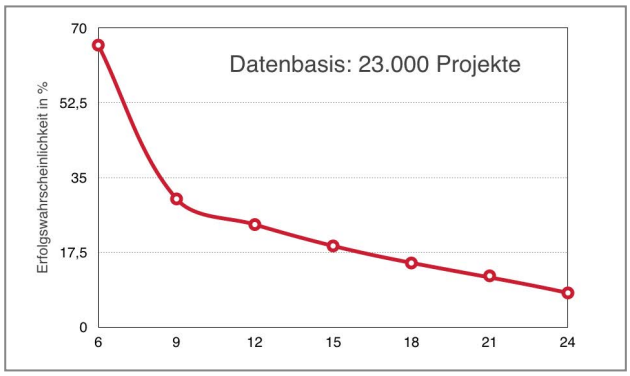

Abb. 11: Studie der Standish Group zeigt: Lange Releases bergen hohes Risiko

Release-Container

Untersuchungen der Standish Group deuten darauf hin, dass Releases, die länger als 6 Monate sind, mit einem sehr hohen Risiko einhergehen (siehe [Griffiths2007]). Die Studienergebnisse visualisiert Abbildung 11.

Wir haben gute Erfahrungen damit gemacht, Releases strikt auf maximal 6 Monate, besser noch 3 Monate zu begrenzen. Bei einem unserer Kunden hat man beispielsweise bei der Scrum-Einführung festgelegt, dass Projekte in Zukunft 3 oder 6 Monate lang sein dürfen.

Dieses Dogma reduziert nicht nur Risiken, es vereinfacht auch die Schätzungen. Jetzt steht das Team nicht mehr vor der Frage, wie lange konkret ein Release dauert. Es muss „nur“ noch die Frage beantworten, ob das Release in einen vorgegebenen Release-Container passt und diese Frage ist viel einfacher zu beantworten.

Beim oben genannten Kunden stellte sich also nur noch die Frage, ob das Projekt in 3 oder in 6 Monaten umgesetzt werden kann. Wenn es länger als 6 Monate dauern würde, wäre dem Unternehmen das Risiko zu groß und sie würden das Projekt nicht durchführen.

Schätzung der Releases

Ob ein Release in einen vorgegebenen Release-Container passt, kann z. B. mit einer Story-Point-Schätzung ermittelt werden (ggfs. in Kombination mit einem Lean-Forecasting-Tool a la [Guesstimate2018]).

Für die nachfolgenden Releases ist dies meist unnötig aufwendig. Diese können z. B. in Relation zu dem ersten Release geschätzt werden, ähnlich wie beim relativen Schätzen mit Story Points. Nehmen wir an, für das erste Release haben wir eine Dauer von 3 Monaten geschätzt. Jetzt überlegen wir, ob das zweite Release ungefähr genauso groß oder eher kleiner oder größer ist. Wenn wir zu dem Schluss kommen, dass es etwas größer, aber nicht doppelt so groß ist, nehmen wie vielleicht 5 Monate für die Dauer an.

Augen auf die Straße

Zielorientierte Roadmaps harmonieren als Planungsinstrument gut mit dem agilen Vorgehen und erlauben eine sinnvolle Einbindung wichtiger Stakeholder wie z. B. Sponsoren oder Lenkungsausschüssen. Die Roadmaps sollten mit einem dazu passenden Controlling-Instrument kombiniert werden, das den Projektfortschritt visualisiert.

Zur den beschriebenen zielorientierten Roadmaps passen Parking Lot Diagramme (siehe [PalmerFelsing2002]). Diese Parking Lot Diagramme zeigen für Features, Feature-Sets oder sonstige Komponenten des Systems den Entwicklungsstand. In unserem Fall sind die dargestellten Einheiten die einzelnen Features aus der Roadmap. Jedes Feature wird durch einen Kasten dargestellt, mit folgender Farbcodierung (Abb. 12):

- weiß: Mit der Entwicklung wurde planmäßig noch nicht begonnen.
- blau: Es wurde mit der Entwicklung begonnen und es gibt keine nennenswerten Probleme.
- grün: Die Entwicklung des Features ist abgeschlossen.
- rot: Es gibt ein Problem bei der Entwicklung.

Wenn ein Feature rot ist und dann doch irgendwann abgeschlossen wird, wird es grün (dass es mal ein Problem ▶

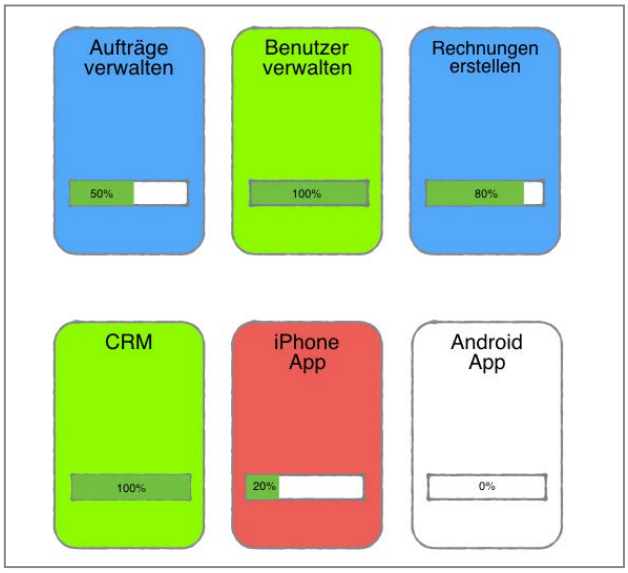

Abb. 12: Projektcontrolling entlang der Roadmap mit Parking Lots

gab, ist jetzt nicht weiter interessant). Ein Feature, dessen Entwicklung noch nicht begonnen hat, kann auch rot werden. Das ist z. B. dann der Fall, wenn aufgrund fehlender Zulieferungen nicht mit der Entwicklung begonnen werden kann, obwohl dies notwendig wäre.

Neben der Farbcodierung zeigt jeder Kasten mit einem Fortschrittsbalken den prozentualen Entwicklungsfortschritt des Features.

Parking Lot Diagrams können außerdem je Feature das geplante Start- und Endedatum angeben. Das Beispiel aus Abbildung 12 verzichtet darauf. Start- und Endedatum können dann sinnvoll sein, wenn es Abhängigkeiten zwischen den Features und mit externen Zulieferern gibt. Wenn diese Abhängigkeiten allerdings massiv sind, stellt man mit einem Parking Lot Diagramm am Ende nur ein GANTT-Chart (siehe [Gantt2018]) anders dar und täuscht somit darüber hinweg, dass man gar nicht so flexibel reagieren kann, wie es im Sinne der agilen Entwicklung notwendig wäre. Man sollte dann die Planung so überarbeiten, dass die Abhängigkeiten reduziert werden.

Inhaltliche Diskussionen führen mit Parking Lot Diagrammen

Der große Charme von Parking Lot Diagrammen im Vergleich zu Release Burnup Charts ist die inhaltlich-fachliche Ausprägung. Der Product Owner kann z. B. mit dem Lenkungsauschuss sinnvoll über den Projektzustand sprechen und Probleme gezielt diskutieren. Das Zentrum der Diskussion sind stets die roten Kästen. Für diese kann jetzt gemeinsam diskutiert werden, was die Ursache für das Problem ist und wie das Problem gemeinsam gelöst werden kann. Die Ergebnisse dieser Diskussion können ganz unterschiedlich sein:

- Vielleicht stellt man fest, dass es ein Problem gibt, weil eine Zulieferung einer anderen Abteilung fehlt. Dann muss der Lenkungsausschuss priorisieren: Ist das Projekt wichtiger als das, woran die zuliefernde Abteilung stattdessen arbeitet? Muss der Lenkungsausschuss bei der zuliefernden Abteilung intervenieren, um die Zulieferungen zu beschleunigen?
- Man könnte entscheiden, das rote Feature zu retten, indem man ein anderes – weniger wichtiges Feature – opfert oder in reduzierter Form implementiert.
- Es könnte sein, dass das Team schlicht langsamer entwickelt, als zunächst angenommen. Dann könnte man untersuchen, ob es organisatorische Hindernisse gibt, der der Lenkungsausschuss beseitigen kann. Oder man könnte beschließen, das Team aufzustocken.
- Natürlich kann man auch gemeinschaftlich zu dem Schluss kommen, dass eine Verschiebung des Termins und aller nachfolgender Termine der Roadmap die beste Lösung ist.

Mit solchen Diskussionen wird die Asymmetrie zwischen Lenkungsausschuss und Product Owner reduziert. Es wird weniger nach dem Motto „Melden macht frei." agiert, sondern gemeinsam nach Lösungen gesucht. Der

Lenkungsausschuss wird stärker in die Verantwortung genommen.

Zusammenfassung

- Story Points, Velocity und Release Burnup Charts sind quantitative Instrumente, die zur Selbstkontrolle im Scrum-Team nützlich sein können.
- Für Diskussion mit Projektexternen (z. B. Lenkungsausschuss) sind hingegen Instrumente notwendig, die inhaltliche Diskussionen erlauben.
- Zielorientierte Roadmaps und Parking Lot Diagramme erfüllen dieses Kriterium.
- Zielorientierte Roadmaps können nahtlos aus Story Maps abgeleitet werden.
- Parking Lot Diagramme lassen sich auf Basis einer zielorientierten Roadmap gut erstellen. ■

Referenzen

[Adzic2012] Gojko Adzic: "Impact Mapping: Making a big impact with software products and projects", 2012.

[PalmerFelsing2002] Stephen R. Palmer, John M. Felsing: „A Practical Guide to Feature-Driven Development", 2002.

[Gantt2018] https://de.wikipedia.org/wiki/Gantt-Diagramm, letzter Besuch: 22.02.2018

[Griffiths2007] Mike Griffiths: „Large Projects Risks", http://leadinganswers.typepad.com/leading_answers/2007/05/large_project_r.html, letzter Besuch: 22.02.2018

[Guesstimate2018] http://www.getguesstimate.com, letzter Besuch: 22.02.2018

[Patton2014] Jeff Patton: "User Story Mapping: Discover the Whole Story, Build the Right Product", 2014.

[Pichler2016] Roman Pichler: „Strategize: Product Strategy and Product Roadmap Practices for the Digital Age", 2016.

STEFAN ROOCK

Stefan Roock zählt zu den agilen Urgesteinen in Deutschland. Schon 1999 führte er erste Projekte mit Scrum und eXtreme Programming durch. Heute berät er Führungskräfte bei agilen Transitionen.
Er ist Autor mehrerer Bücher zu agilen Themen und regelmäßiger Sprecher auf Konferenzen und Tagungen.
Er lebt mit seiner Frau, zwei Söhnen und einer Tochter in Geesthacht und schreckt auch im Dezember nicht davor zurück, sich beim Windsurfen in der Ostsee nasse Füße zu holen.

IMPRESSUM

Chefredaktion (verantwortlich):
Dr. Wolf-Gideon Bleek

Redaktionsadresse:
it-agile GmbH, Willy-Brandt-Straße 1, 20457 Hamburg
www.it-agile.de, info@it-agile.de

Gerichtsstand und Erfüllungsort: Hamburg

Layout: Jasna Wittmann Kommunikationsdesign
www.jasnawittmann.de

Gestaltung des Titelfotos: Henriette

Autoren: Marco Bonk, Sven Günther, Sebastian Keller, Holger Koschek, Lars Lentfer, Anna-Lena Lorenz, Ilja Preuß, Robert, Kathrin Schröder, Stefan Roock, Mathias Schröder, Nadine Wolf

Druck: Drucktechnik Altona
Große Rainstraße 87, 22765 Hamburg
www.drucktechnik-altona.de

Bildnachweise: S. 36, 54: istockphoto.com
S. 14, S. 42: stock.adobe.com
S. 22: Pixabay

Alle anderen Fotos und Grafiken: it-agile GmbH