Gemeinsames Systemverständnis durch User Story Mapping

Thema der wissenschaftlichen Vertiefung von Julian Alt

Inhaltsverzeichnis

1 User Story Mapping

Die agile Entwicklung spaltet die Anforderungen an ein System, die zum Projektstart unvollständig vorliegen, aber im Laufe des Projekts ergänzt werden, in Anforderungen an kleine Teile des Systems auf. Alle Anforderungen dürfen laufend aktualisiert werden, bis auf die Anforderungen an das aktuell zu erzeugende Teil.

Die Anforderungen liegen in Scrum als sogenannte Backlog Items im Product Backlog des Product Owner. Das Product Backlog enthält also die Forderungen an das zu realisierende System. Bei den agilen Ansätzen werden die Anforderungen oft durch User Stories[1] dargestellt. Die **Aufspaltung der Anforderungen** erlaubt es, dass eine Komponente nach der anderen realisiert wird und dass die Entstehung des Systems gemeinsam von Entwicklern und Kunden beobachtet und gesteuert werden kann. Entwickler und Kunde bestimmen dabei gemeinsam, was als nächstes programmiert werden soll.

Jeff Patton hat mit seiner Technik „User Story Mapping" eine **Visualisierungsmethode** entwickelt, die es erlaubt, das „flache" Product Backlog strukturiert darzustellen. Hierzu werden die **Karten der einzelnen User Stories**, die Story Cards, eines Backlogs **visuell geschickt** (etwa an einer Tafel) so **angeordnet**, dass diese Anordnung der Karten einen Überblick über die Funktionalitäten des zu realisierenden Systems gibt. Martin Fowler sagte hierzu im Jahre 2014 in [1]:

„Story mapping is a technique that provides the big picture that a pile of stories so often misses."

Mit der Technik des User Story Mapping soll vermieden werden, dass in inkohärenter Weise eine Reihe von lauffähigen Code-Inkrementen programmiert wird, die zusammen eben nicht das gewünschte System ergeben.

1.1 Ziele des User Story Mapping

Das visuelle Bild eines Systems in Form geordneter Karten erlaubt:

- eine **bessere Kommunikation** mit dem Nutzer und im Entwicklungsteam,
- eine **bessere Übersicht** über die **Kundenbedürfnisse** und das **zu bauende System** als ein loser Stapel von User Stories und
- das **Vermeiden von Features**, die dem Kunden am Ende **nicht den gewünschten Nutzen** bringen**.**

Dabei ist **User Story Mapping** eine **Brücke**, um das **Design** eines Systems in die **Programmierung** zu bringen. Man kann eine User Story Map auch als eine Verbindung zwischen den **Nutzerexperten** – sie denken gerne in Nutzerfunktionalitäten, also etwas Grobem, und den **Implementierern** – sie denken in Programmcode, also etwas Feinem, sehen. Letztendlich ist es das Ziel, durch die Visualisierung der Story Map ein **gemeinsames Systemverständnis zwischen Kunden und Entwicklern** zu erzeugen.

[1] Bei Extreme Programming (XP) ist es Pflicht.

Ein **gemeinsames Verständnis** ist nicht allein mit Hilfe der Anfertigung von Dokumenten zu erreichen, da Dokumente von ihren Lesern auf unterschiedliche Weise interpretiert werden können. Viel leichter erreicht man ein gemeinsames Verständnis durch Gespräche und Diskussionen, also durch **gemeinsame Interaktionen zwischen Scrum Team und Kunden** inklusive der Dokumentation der Ergebnisse dieser Interaktionen mithilfe von Bildern, Videos, Karteikarten oder Klebezetteln. Es geht nicht darum, gute Stories zu schreiben, vielmehr geht es darum, das zu tun, was eine Story in ihrem eigentlichen Sinn machen soll, nämlich eine Geschichte aus der Sicht eines bestimmten Users zu erzählen, um ein gemeinsames Verständnis zu erzeugen.

User Story Mapping strukturiert den Stapel von User Stories und damit die Elemente der Kommunikation zwischen Entwickler und Nutzer. Die User Story Map ergänzt das Product Backlog um eine Struktur. Das Product Backlog soll also nicht eine flache Struktur von Backlog Items, oft User Stories, sein. Es soll visuell in strukturierter Weise dargestellt werden,

- zum einen, damit durch **gemeinsame Diskussionen** mit dem Kunden und den Entwicklern ein **gemeinsames Wissen** im Projekt entsteht und
- dass dieses Wissen durch seine **Struktur** die **Gesamtsicht des Systems** darstellt.

User Story Mapping ist also eine Technik, ein **gemeinsames Verständnis für ein System** zu erzeugen. Dazu muss die User Story Map für alle einsehbar sein („**shared workspace**“). Änderungen erfolgen im Dialog. Die Kartentechnik erlaubt es aber, dass die Vorstellung und Diskussion einzelner User Stories im Team nicht unterbrochen werden muss, falls die Zuhörer währenddessen ergänzende Ideen haben. Fallen den Zuhörern während eines Vortrags weitere Ideen ein, so notieren sie ihre Ideen in Stichworten auf Karten und diskutieren ihre Ideen erst nach dem gerade laufenden Vortrag mit dem Vortragenden und anderen Anwesenden.

1.2 Aufbau einer User Story Map

In Abbildung 1-1 sei beispielhaft der Aufbau einer User Story Map dargestellt:

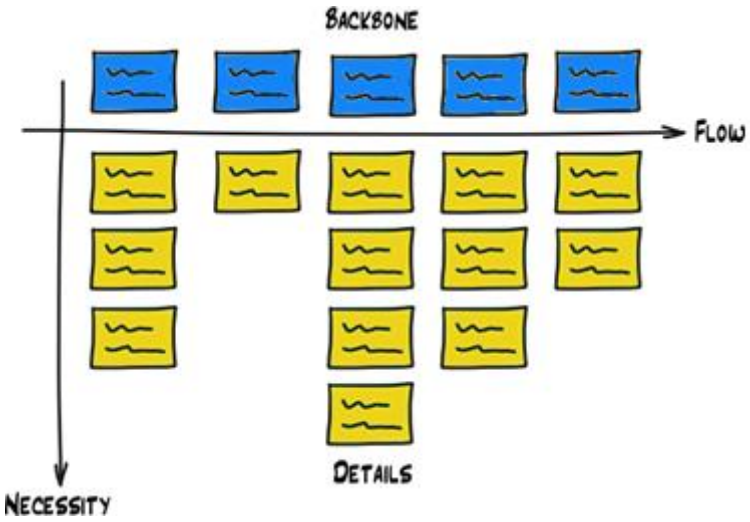

Abbildung 1-1 Prinzipieller Aufbau einer Story Map

Im Folgenden die Erklärung dieser Abbildung:

- In der oberen Reihe, dem sogenannten **Backbone** werden die **Nutzerfunktionalitäten** aufgereiht, die einen Wert für einen Kunden haben. Jede Karte beschreibt eine **Aktivität** (engl. **activity**) aus der Sicht eines bestimmten Benutzers, die mit Hilfe des Systems ausgeführt werden soll.
- Die Anordnung von links nach rechts entspricht dem Fluss (engl. Flow) der Story, welcher einen möglichen **Erzählfluss** einer Geschichte widerspiegelt. Ungeachtet der Reihenfolge der Aktivitäten ist es wichtig, dass durch den visualisierten Fluss der zusammengesetzten Geschichte eine Gesamtübersicht bezüglich des zu entwickelnden Systems geschaffen werden soll. Selbstverständlich können verschiedene Nutzerfunktionalitäten im System unabhängig voneinander aufgerufen werden. Wichtig ist, dass der Backbone einen groben **Überblick über das System** gibt, indem es eine Möglichkeit beschreibt, wie ein bestimmter Anwender das System verwenden würde.
- Die **Story-Cards** unterhalb des Backbones entsprechen **Tätigkeiten** (engl. **tasks**), die dem Nutzer dabei helfen sollen, ein Ziel zu erreichen. In Bezug auf die Story Map sollen die Tätigkeiten (tasks) dazu dienen, eine Aktivität des Backbone durchführen zu können. Sie entsprechen damit einer detaillierten Form der Aktivitäten aus dem Backbone.

Damit bilden die User Story Maps die Funktionalität des aktuell bekannten, zu entwickelnden Systems ab. Ungeachtet der Reihenfolge der Aktivitäten ist es wichtig, dass durch den visualisierten Fluss der zusammengesetzten Geschichte eine Gesamtübersicht bezüglich des zu entwickelnden Systems geschaffen werden soll.

1.3 Aktivitäten oder Nutzerfunktionalitäten

Eine jede zu durchlaufende Aktivität kann nun in einzelne Schritte zerlegt werden, die innerhalb dieser Aktivität ablaufen. Diese Schritte werden von links nach rechts aufgemalt, während die Details jedes Schritts vertikal nach unten verfeinert werden. Abbildung 1-1 zeigt hierfür ein Beispiel.

Die entscheidenden Faktoren, die ein Produkt formen und den Kontext bilden, hängen oft über der Map wie **Produktziele**, **Informationen über Nutze**r oder **weitere Ideen**.

Jede Tätigkeit kann feiner granuliert werden, indem die Verfeinerung in detaillierterer Form unterhalb der groben Tätigkeit angeordnet wird. Hierbei steht die vertikale Anordnung der Tätigkeiten aber auch für eine Priorisierung, wobei höher angeordnete Tätigkeiten eine höhere Priorität haben als niedriger angeordnete Tätigkeiten.

1.4 Ein konkretes Beispiel

Im folgenden Kapitel soll beispielhaft eine Story Map für ein einfaches E-Mail-Verwaltungssystem entworfen werden. Im ersten Schritt werden alle Ideen gesammelt und mit allen Projektbeteiligten diskutiert. Das Ergebnis ist eine Sammlung von Aktivitäten

und Tätigkeiten, die durch das Programm unterstützt werden sollen. Die folgende Abbildung zeigt, wie ein Ergebnis dieses Brainstormings aussehen könnte[2]:

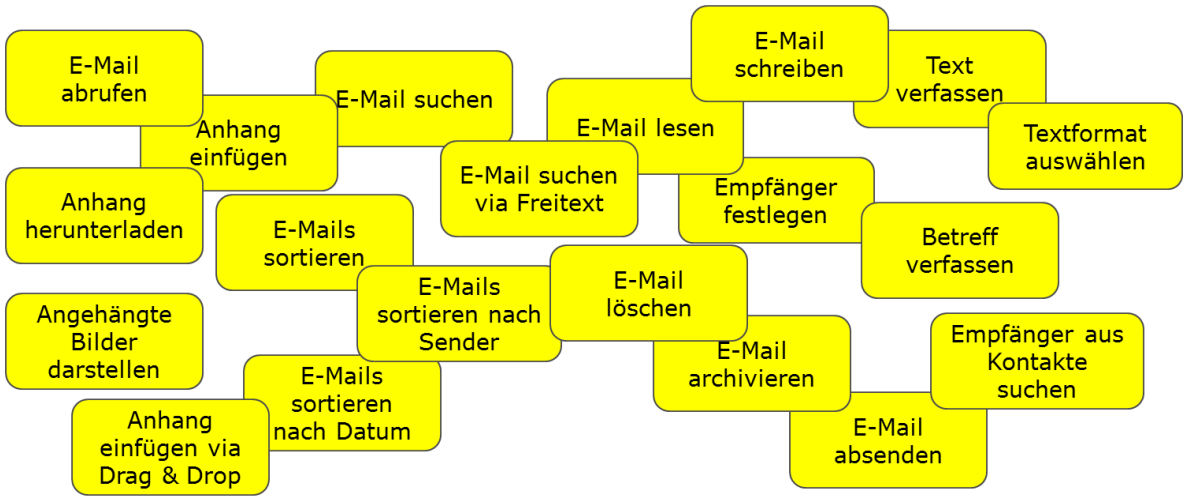

Abbildung 1-2 Ergebnis des Brainstorming für ein einfaches E-Mail-System

Jede Karte entspricht einer User Story. Jede dieser Stories kann nun in das Product Backlog übernommen werden. Um das Product Backlog in eine übersichtliche Struktur zu bringen, wird die Gesamtheit der User Stories in die Form einer User Story Map gebracht. Die User Stories werden zu **Hauptfunktionalitäten** zusammengefasst, die das sogenannte **Backbone** darstellen. Es sind (allgemeine) Aktivitäten, die ein Benutzer des Systems durchführen will. Die Reihenfolge von links nach rechts beschreibt den Erzählfluss. Die darunter liegenden User Storys sind detailliertere Aktivitäten, die notwendig sind, um die Hauptfunktionalitäten erledigen zu können. Die Anordnung von oben nach unten beschreibt die Priorität und den Detailgrad einer Aktivität. Abbildung 1-3 zeigt eine Möglichkeit der Anordnung der User Stories aus obiger Abbildung zu einer User Story Map:

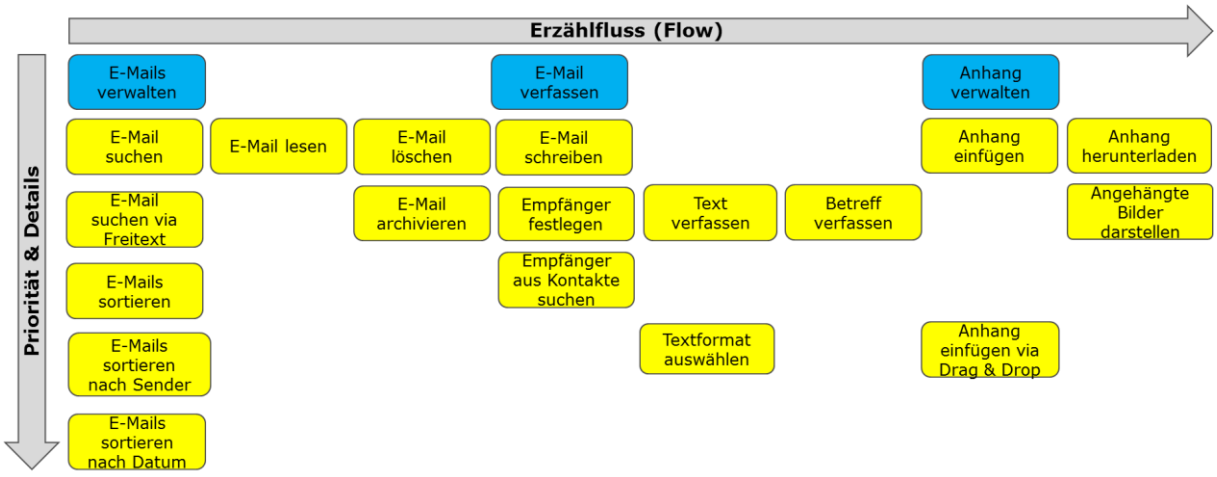

Abbildung 1-3 Story Map eines einfachen E-Mail-Systems[3]

[2] Dieses Beispiel soll der Übersicht halber nur einen Ausschnitt der möglichen Aktivitäten zeigen. Üblicherweise müssen weitaus mehr Aktivitäten gefunden werden, um ein Gesamtsystem zu beschreiben.

Der Erzählfluss des Backbones könnte wie folgt aus der Story Map interpretiert werden: „Ein Benutzer eines E-Mail-Programms möchte seine E-Mails mithilfe des Systems verwalten können. Anschließend möchte er eine E-Mail verfassen und diese mit einem Anhang versehen.“ Als Verfeinerung der Aktivität „E-Mails verwalten“ soll der Benutzer zum Beispiel eine E-Mail suchen können. Eine detailliertere – und darum darunter angeordnete – Aktivität wäre, eine E-Mail via Freitext suchen zu können.

Ein möglicher Flow – also von links nach rechts angeordnet – wird hier so beschrieben, dass ein Benutzer des E-Mail-Programms zum Verwalten seiner E-Mails eine E-Mail erst suchen, dann lesen und anschließend löschen kann. Hierbei sei bemerkt, dass auch diese Tätigkeiten zur Realisierung feiner granuliert werden sollten, aber zur besseren Übersicht in diesem Beispiel darauf verzichtet wurde.

Ist eine Story Map erstellt, gibt es nun verschiedene Techniken, wie die Aktivitäten unterhalb des Backbones strukturiert werden können. Kapitel 1.7 soll einige dieser Techniken erläutern. Zuvor soll aber noch auf die Begriffe **Minimum Viable Product**, **Minimal Marketable Release** sowie den Unterschied zwischen **Output** und **Outcome** eingegangen werden, da diese Begriffe essenziell bei der Anwendung dieser Techniken sind.

1.5 Minimum Viable Product und Minimal Marketable Release

Der Begriff Minimum Viable Product (kurz MVP) hat seinen Ursprung in den Lean Startup-Prinzipien[4]. Er wurde im Jahr 2001 von Frank Robinson[5] geprägt und wurde populär durch Steve Blank [2] und Eric Ries [3].

Im Prinzip dreht sich der Ansatz des Lean Startup darum, mit geringem Startkapital möglichst effizient möglichst viel Geld zu verdienen und ein **unnötiges Risiko zu vermeiden**. Es geht darum, eine **Produktidee** experimentell möglichst schnell zu testen, um direkt aus dem Kundenfeedback lernen zu können. Dies kann man teilweise auch bewerkstelligen, ohne dass überhaupt irgendwelcher Code geschrieben wird.

Die Idee des **Minimum Viable Product** bedeutet, dass es absolut vermieden werden soll, dass ein Produkt aufwendig realisiert wird, dann aber keinen Markt findet. Es soll hingegen Zeit, Arbeit und Geld gespart werden. Durch den **iterativen Prozess** des Validierens und des Lernens (engl. **„validate and learn“**) wird zu einem möglichst frühen Zeitpunkt das richtige Produkt Stück für Stück gefunden.

Das folgende Bild zeigt die iterative Entwicklung des MVP nach Lean Startup-Prinzipien:

---

[3] Aus Gründen der Lesbarkeit konnten die Funktionen „E-Mail absenden“ und „E-Mail abrufen“ nicht dargestellt werden.

[4] Das Lean Management (dt. „schlankes Management“) ist ein Managementprinzip und definiert Denkprinzipien, Methoden und Herangehensweisen, um Prozesse zur Herstellung industrieller Güter entlang der gesamten Wertschöpfungskette effizient zu optimieren. Mit Lean Startup-Prinzipien wird die Anwendung von Lean Management-Ansätzen in Bezug auf Startup-Unternehmen beschrieben.

[5] CEO der Firma SyncDev, http://www.syncdev.com/

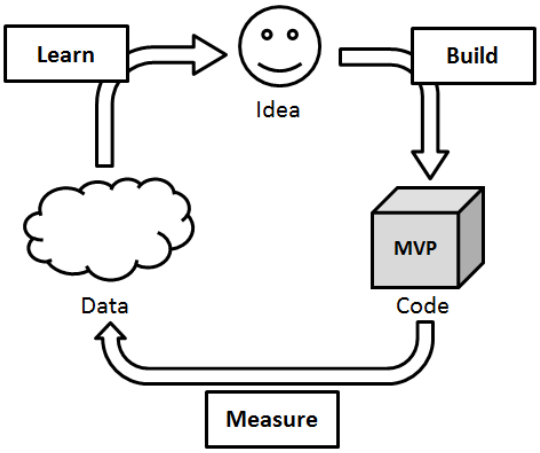

Abbildung 1-4 Iterative Entwicklung des MVP nach Lean Startup-Prinzipien

Das MVP eignet sich besonders gut, um **Risiken abschätzen und minimieren** zu können. Zum Austesten einer **Produktidee** reicht ein einfacher Mock-up, der sogenannten "early adopters", also bestimmten Kunden, die als Trendsetter dienen, repräsentiert wird.

Für einen möglichst frühen **Markteintritt** reicht ein kleiner **Prototyp**, mit dem evaluiert werden kann, ob das Produkt am Markt ankommt. Der Begriff für einen ersten Stand, den man wirklich vermarkten und nicht nur mit einer kleinen Anzahl erster Kunden testen will, ist das **Minimal Marketable Release** (**MMR**).[6] Das MMR ist damit in der Regel kein MVP – also kein Experiment – mehr, sondern deutlich größer – ein marktfähiges Release.

Für das MMR gilt der Ansatz „**Weniger ist mehr“**. Ein Produkt soll eine **minimale Anzahl an Features** unterstützen, sodass das Produkt **erfolgreich die Erwartungen seiner Anwender** erfüllt. Jedes nicht unbedingt notwendige Feature wird im Konzept für das zu entwickelnde Produkt weggelassen.

Das MVP kann dafür genutzt werden, um das MMR zu bestimmen. Auch hier spielt der Prozess des Validierens und des Lernens eine fundamentale Rolle:

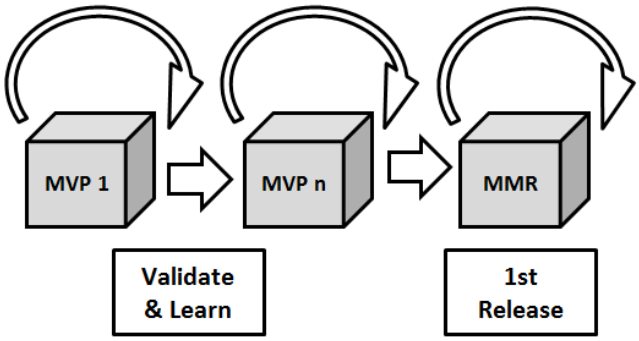

Abbildung 1-5 Nutzen des MVP zur Identifikation des MMR

[6] Häufig wird fälschlicherweise das MVP als MMR gesehen.

Eine User Story Map kann dazu dienen, sowohl frühe Experimente, also MVPs, als auch MMRs zu identifizieren. Wichtig dabei ist der ständige Hintergedanke, mit einem möglichst minimalen Einsatz an Risiko genau das zu entwickeln, was der Kunde erwartet.

1.6 Unterschied zwischen „Output“ und „Outcome“

Um bei der Softwareentwicklung das Minium Viable Product beziehungsweise das Minimal Marketable Release zu finden, ist nach Jeff Patton [1] zwischen den Begriffen **Output** und **Outcome** zu unterscheiden:

- Als **Output** wird alles beschrieben, was während einer Softwareentwicklung erstellt wird. Darunter fallen Spezifikationen, Anforderungen, Code und das Produkt selbst.
- Mit **Outcome** wird das beschrieben, was ein entwickeltes Produkt bei seinen Anwendern bewirkt. Es beschreibt den effektiven **Nutzen** und wie es den Anwender in seiner ausübenden Tätigkeit beeinflusst.

Es ist wichtig, dass das Minimum Viable Product beziehungsweise das Minimum einen möglichst hochwertigen **Nutzen** (**Outcome**) bewirkt, sodass das Produkt von seinen Anwendern positiv angenommen und benutzt wird. Ein **langfristiger Nutzen**, den die Verwendung der entwickelten Software mit sich bringt, wird von Patton als **Impact** bezeichnet, der sich beispielsweise auf den Umsatz einer Firma, in der die entwickelte Software verwendet wird, auswirkt. Dies wiederum bewirkt für den Softwareentwickler beziehungsweise die Entwicklungsfirma der Software eine Verbesserung der Verkaufszahlen. Das **Outcome** und der Impact sollen folglich möglichst vergrößert werden, während der **Output**, d.h. die Zahl der Erzeugnisse der Software-Entwickler, schlank gehalten werden soll, da im Vordergrund die effektive Lösung von Problemen (also der Nutzen) stehen soll und nicht die Tatsache der Entwicklung eines Softwareprodukts.

Die folgende Abbildung soll den Unterschied zwischen den Begriffen Output und Outcome verdeutlichen:

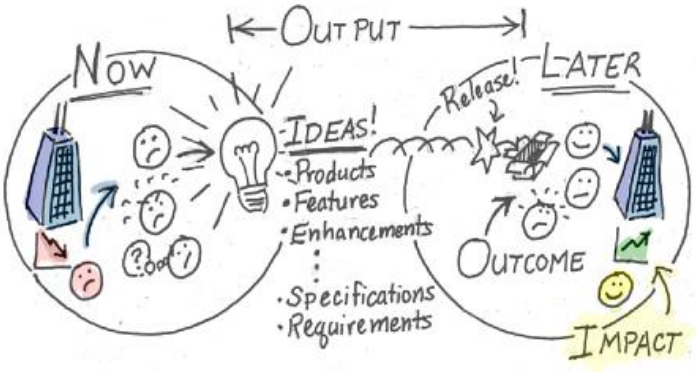

Abbildung 1-6 Unterscheidung der Begriffe Output und Outcome[7]

[7] Quelle: [1]

1.7 Techniken

In den folgenden Kapiteln 1.7.1 bis 1.7.3 werden Techniken bei der Verwendung von Story Cards in einer Story Map beschrieben.

1.7.1 Einsatz von Prioritäten

Die **Typen der Nutzer des Systems** und **wie sie das System nutzen würden**, muss in eine **Proritätsreihenfolge** gebracht werden. In der Regel werden zu viele Aktivitäten und Tätigkeiten gefunden, um sie in einem Release unterzubringen. Ein möglicher Ansatz ist, die Nutzer des Systems zu priorisieren und zuerst anhand des wichtigsten Nutzers eine Story Map zu erstellen. Findet sich anschließend noch Platz für Aktivitäten oder Tätigkeiten von Nutzertypen mit niedrigerer Priorität, können diese in die Story Map eingepflegt werden.

1.7.2 Planung von Releases

Eine erste Möglichkeit zur Verfeinerung der Story Map entsteht aus der Tatsache, dass in der Regel erst einmal zu viele Aktivitäten und Tätigkeiten dokumentiert werden, als dass all diese bis zum nächsten Release implementiert werden können. Das Ziel muss es sein, das Minimal Marketable Release (siehe Kapitel 1.5) für ein Release zu finden. Um das MMR, also das kleinstmögliche (lauffähige) Produkt, das erfolgreich die gewollte, minimal sinnvolle Leistung eines Systems erbringt, aus der Story Map zu filtern, wird diese in horizontale Ebenen durch Linien, die ein Release vom anderen abgrenzen, aufgeteilt.

Abbildung 1-7 soll die Aufteilung einer Story Map nach Releases darstellen:

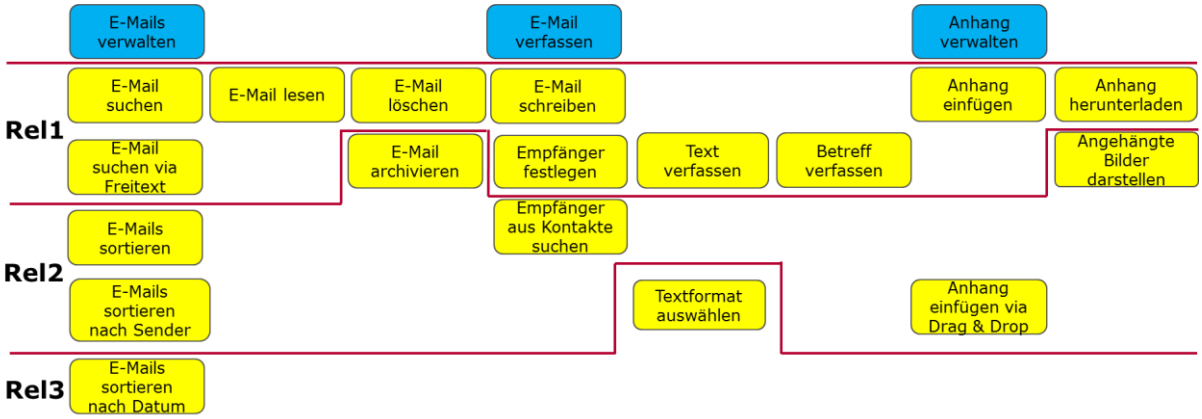

Abbildung 1-7 Visualisierung einer Release Roadmap mithilfe einer Story Map

Jede Ebene sollte einem MMR für ein Release entsprechen.

Die erste Ebene liegt analog zur entsprechenden Priorität der Story Map ganz oben am Backbone. Dabei muss geplant werden, welche Aktivitäten und Tätigkeiten minimal notwendig sind, um dem erwartetem Nutzen des Systems zu genügen.

Von dem ersten Release übrig gebliebene User Stories werden auf die Releases darunter übertragen und wiederum gefiltert, bis alle Tätigkeiten realistisch für jeweils

ein einziges Release aufgeteilt sind. Die oberste Ebene der übrig gebliebenen User Stories entspricht im Optimalfall dem MMR, das für das anstehende Release entwickelt werden soll.

Releases können in diesem Fall sowohl für interne, als auch externe Releases stehen. Für interne Releases kann man auch von der Entwicklungsstrategie für den Bau eines Release mithilfe einer Story Map sprechen, wie im nachfolgenden Kapitel erläutert werden soll.

1.7.3 Entwicklungsstrategie für den Bau eines Release

Nachdem die umzusetzenden User Stories nun dem MMR für jeweils ein Release entsprechen, kann davon ausgegangen werden, dass mit Sicherheit auch nur genau das realisiert wird, was den Kunden am Ende zufrieden stellt.

Von diesem Zeitpunkt an kann die Story Map in eine Sprint-Planungstafel (engl. Sprint Planning Board) umgewandelt werden[8]. Als erstes muss eine erwünschte Wirkung (**„desired outcome“**, vgl. Kapitel 1.6) für das im Sprint zu erstellende Inkrement definiert werden. Im nächsten Schritt gilt es, diejenigen minimal notwendigen aus den bestehenden Tätigkeiten zu entnehmen, die dazu beitragen, das gewünschte Ziel zu erreichen. Diese werden an oberster Stelle, also mit höchster Priorität, am Story Board angebracht. Auch hier kann und soll bewusst vom Minimum Viable Product in Bezug auf ein Inkrement gesprochen werden. Die höher priorisierten Tätigkeiten sollten genau den minimal umzusetzenden Tätigkeiten entsprechen, die benötigt werden, um die gewünschte Wirkung eines Inkrements zu erfüllen. Mithilfe dieser Technik entsteht mit jeder Iteration, also jedem Sprint, ein immer genaueres, lauffähiges Abbild des Endprodukts.

Abbildung 1-8 zeigt die Aufteilung einer Story Map in Sprints nach einer gewählten Entwicklungsstrategie für ein erstes Release aus dem Beispiel des E-Mail-Systems:

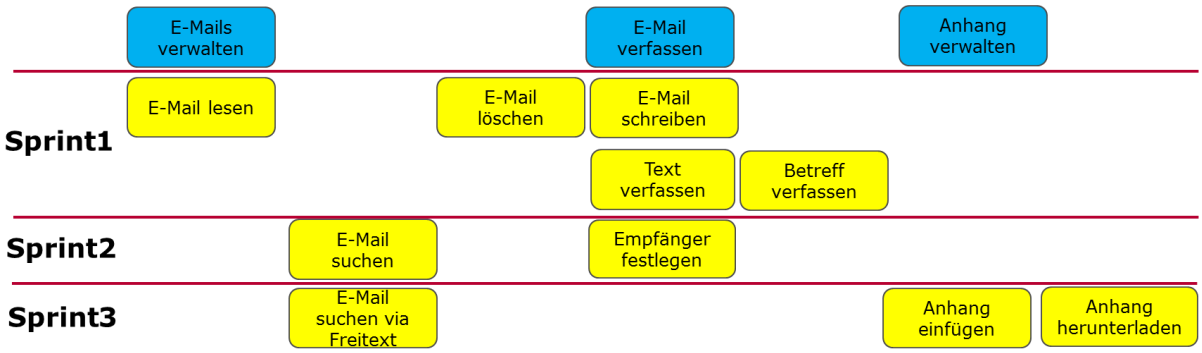

Abbildung 1-8 Story Map mit priorisierten Tätigkeiten für ein Release

[8] Natürlich kann auch eine separate Sprint-Planungstafel erstellt werden, nur sollten die Tätigkeiten nicht physisch vom „Big Picture“, also der Story Map entfernt werden, um die Gesamtübersicht zu wahren.

1.8 Bewertung

Jeff Patton hat mit seiner Technik des User Story Mappings eine Herangehensweise entwickelt, um das Ziel eines **übergreifenden gemeinsamen Verständnisses für ein Produkt mit Hilfe von Visualisierung und Kommunikation** zu erreichen. Die Anordnung von User Stories zu einer User Story Map kann sowohl dabei helfen, Releases zu planen, als auch die innerhalb von Sprints zu implementierende User Stories zu koordinieren. Im Mittelpunkt steht hierbei zudem jeweils der Gedanke an das **Minimum Viable Product** bzw. das **Minimal Marketable Release**, das sowohl die Entwickler selbst vor Selbstüberschätzung und daraus resultierender Überlastung schützen soll, als auch sicherstellt, dass ein Release am Ende genau das macht, wofür es ursprünglich geplant war, und damit seinen erhofften Nutzen bringt. In Bezug auf Scrum lassen sich Iterationen, in denen die Story Map verfeinert, ausgebessert und so für den nächsten Sprint vorbereitet wird, im so genannten **Product Backlog Refinement**[9] – auch Backlog Grooming genannt – durchführen.

Patton stellt in seinem Buch für spezielle Projekte individualisierte Beispiele zur Anwendung von User Story Maps vor. Die Anwendung von User Story Maps kann und soll für unterschiedliche Projekte unterschiedlich individualisiert werden. Reines Nachahmen führt in den wenigsten Fällen zum Erfolg. Man soll nicht "kupfern"! Darüber hinaus ist mit dem alleinigen Aufbau einer Story Map der Erfolg eines Projekts noch nicht gegeben. Der gesamte Entwicklungsprozess – also beispielsweise die Anwendung eines agilen Vorgehensmodells – muss in seiner Art auch gelebt werden: Ohne eine ständige Kommunikation zwischen allen Projektbeteiligten, um ein gemeinsames Verständnis zu erreichen, ist das Konzept agiler Entwicklungsmethoden schlichtweg verfehlt. Die Anwendung der von Jeff Patton vorgestellten Methoden zum Erstellen einer Story Map schaffen den Übergang von einem flachen, unübersichtlichen Product Backlog zu einem strukturierten Fundament eines zu entwickelnden Produkts, das allen Projektbeteiligten ein gemeinsames Verständnis bezüglich der Fragen „was, wofür und für wen“ ermöglicht.

[9] Product Backlog Refinement wurde in der ursprünglichen Version des offiziellen Scrum Guides von Jeff Sutherland und Ken Schwaber nicht explizit erwähnt, wurde aber in der aktuellen Version [4] hinzugefügt. Es dient dem Product Owner und dem Entwicklungsteam dazu, das Product Backlog am Ende eines Sprints zu verfeinern und zu überarbeiten, um auf den nächsten Sprint vorbereitet zu sein.

2 Literaturverzeichnis

[1] J. Patton, User Story Mapping - Discover the whole story, build the right products, O'Riley, 2014.

[2] S. Blank, The Startup Owner's Manual: The Step-By-Step Guide for Building a Great Company, K&S Ranch, 2012.

[3] E. Ries, The Lean Startup: How Constant Innovation Creates Radically Successful Businesses, Portfolio Penguin, 2011.

[4] J. Sutherland und K. Schwaber, „The Scrum Guide (pdf),“ 07 2013. [Online]. Available: http://www.scrumguides.org/. [Zugriff am 13 03 2015].