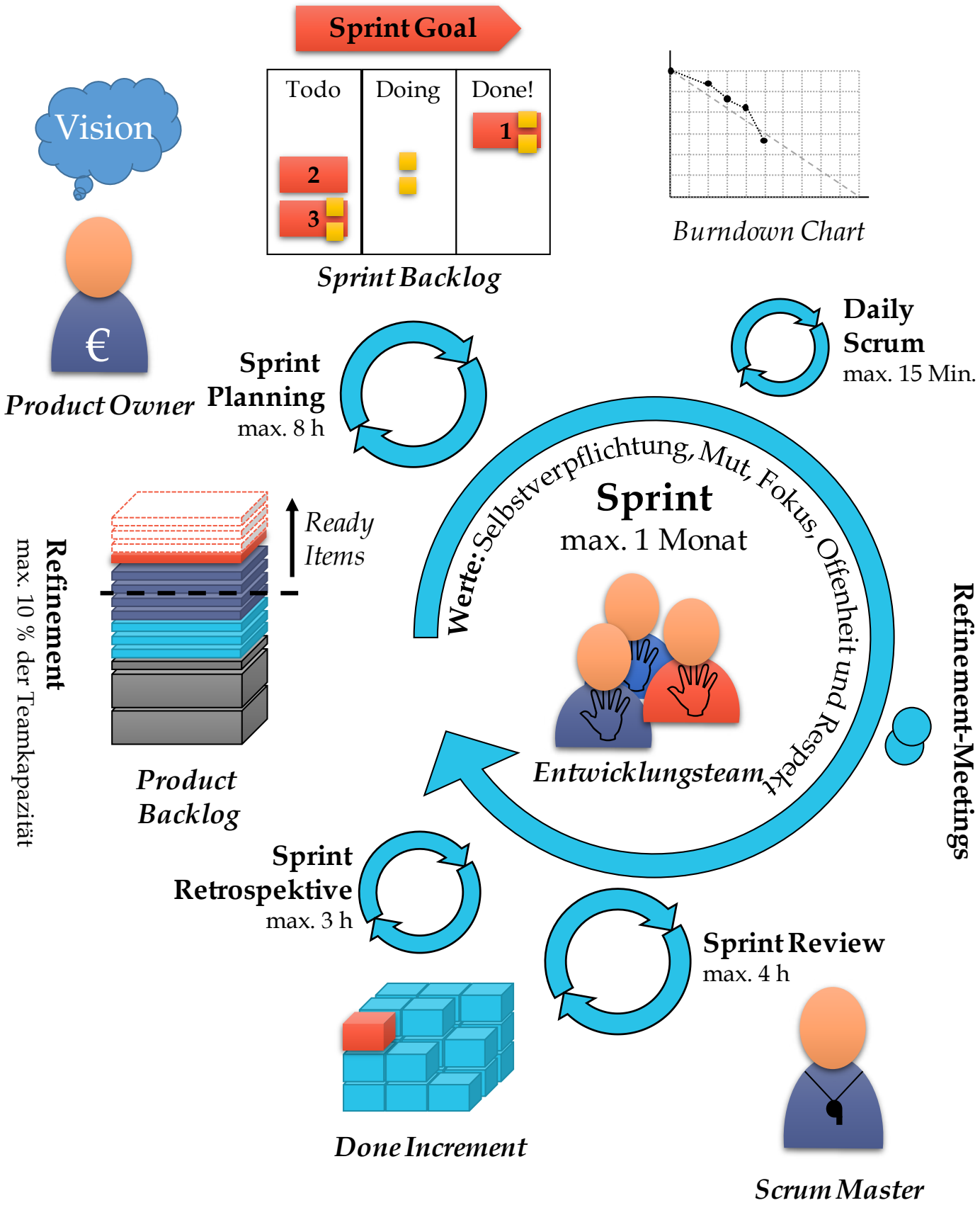

Einstieg in die Agilität

Scrum und Kanban im Überblick

Die Common Sense Team GmbH hilft Unternehmen und Institutionen in der Verwaltung dabei, agiler zu arbeiten. Beispiele:

- Einführen von agilen Strukturen (Scrum, Scrum @ Scale)
- Bessere Zusammenarbeit durch eine gemeinsame Dokumentenablage (DMS, E-Akte)
- Strategische Planung durch Szenarioarbeit
- Ausbildung von neuen Mitarbeiter*innen, Ausbildung von Fachkräften (eduScrum, TWI)

Ihr Unternehmen, Ihre Institution ist wichtig, für die Menschen, die Ihre Produkte kaufen oder Dienste nutzen. Für die Menschen, die dort ausgebildet werden und arbeiten.

Nehmen Sie Kontakt mit uns auf, wenn Sie Fragen haben oder mehr wissen wollen.

Kontakt

Common Sense Team GmbH
Kaiserstraße 209
76133 Karlsruhe
Telefon: 0721/57034-65

E-Mail: info [bei] commonsenseteam.de
www.commonsenseteam.de
Blogs: www.teamworkblog.de, https://agile-verwaltung.org/

Inhaltsverzeichnis

1 Überblick

Lean Thinking bedeutet, Verschwendung zu vermeiden, ohne den Kunden zu frustrieren. Agilität ist wie Lean Thinking, nutzt aber zusätzlich Feedback, um Produkte und Abläufe zu verbessern. Diese Unterlage macht Sie mit wichtigen Konzepten, Begriffen und Praktiken vertraut. Viele dieser Begriffe sind vor allem in IT-Organisationen populär. Sie lassen sich aber auf viele andere Bereiche übertragen.

Viele Organisationen haben sich in den letzten Jahren sehr stark spezialisiert. Was einst ein Vorteil gewesen sein könnte, wird heute zum Problem, wenn man neue Lösungen (Produkte oder Services) verteilen und testen will, denn …

- … neue Lösungen sind nicht nutzbar, wenn sie nicht zur Verfügung gestellt werden.
- … bereit gestellte Lösungen nützen nichts, wenn die Anwender sie nicht benutzen.
- … benutzte Lösungen bringen keinen Vorteil, wenn sich die Arbeitsweisen der Anwender nicht ändern.
- … neue Lösungen, neue Software per se haben keinen Wert. Nur geänderte Arbeitsweisen bringen Geld ein.

Die folgende Abbildung 1 zeigt die Themen im Überblick.

- Herausfinden, was der Kunde braucht: Design Thinking, Business Model Generation, Lean Startup
- Neue Produkte entwickeln: Scrum, Kanban, Concurrent Engineering
- Services betreiben: Kanban, Continuous Delivery, Lean Production/Lean Thinking

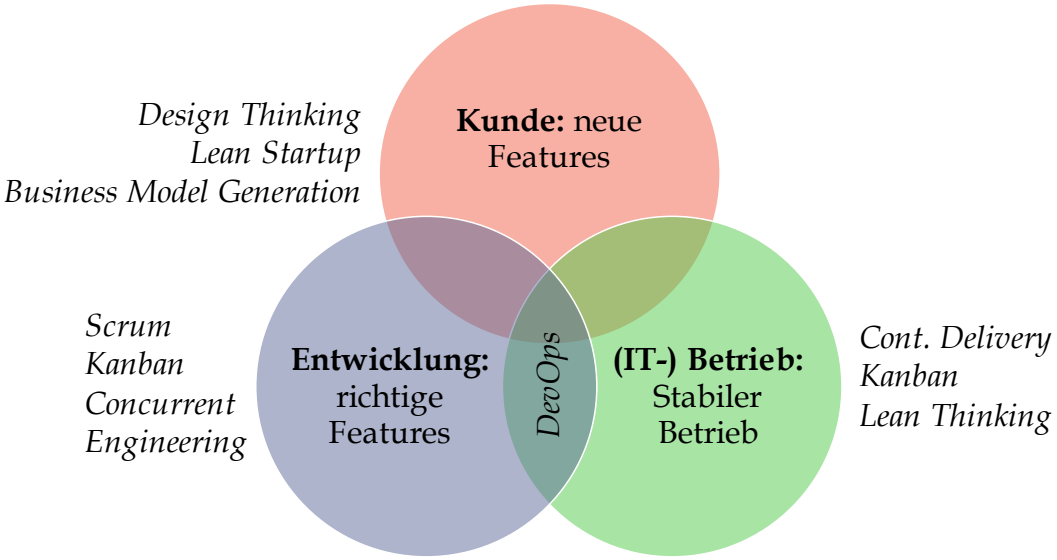

Abbildung 1: Themen im Überblick

Was soll dieses Training erreichen?

- Agile Formen der Zusammenarbeit und Selbstorganisation erleben und verstehen
- Erlernen der wichtigsten Begriffe von Scrum und Kanban

- Wie kann man schneller neue Produkte entwickeln, die einen Mehrwert für Kunden haben?
- Wie kann man im Betrieb sparen?

Dieses Training ist keine Verkaufsveranstaltung für Scrum oder Lean Thinking. Es soll beim Verstehen helfen, sodass Sie Ihre eigene Entscheidung darüber treffen können, ob Sie sich weiter mit Agilität und Lean Thinking beschäftigen wollen.

2 Historischer Überblick

Viele denken, dass Agilität die neue Sau ist, die gerade durch das Dorf getrieben wird. Scrum und Agilität nutzen viele Ideen aus den letzten 200 Jahren. Daher lohnt sich ein kurzer Blick auf die Geschichte.

Scrum und Kanban bauen beide auf Lean Thinking und den Erfahrungen bei Toyota auf[1]. Die Ideen bei Toyota wurden nach dem 2. Weltkrieg aus den USA importiert. Dort gab es ein Ausbildungsprogramm (TWI – Training within Industry) für Firmen, deren Mitarbeiter zum Militärdienst eingezogen wurden. Dieses Programm sollte dabei helfen, schnell neue Mitarbeiter anzulernen, Konflikte zu vermeiden und Arbeitsabläufe zu optimieren.

Hier sind wir an einem Punkt, der für das Grundverständnis sehr wichtig ist. Wie können Sie eine industrielle Produktion organisieren, wenn erfahrene Mitarbeiter und Material fehlen? Sie müssen mit den Leuten und den Dingen auskommen, die zur Verfügung stehen. Der Supervisor war damals dafür verantwortlich, dass neue Mitarbeiter angelernt, Konflikte vermieden und ständig etwas verbessert wurde.

Mit dem Anlernen konnte man sich nicht ewig Zeit lassen. Deswegen musste man ein Verfahren entwickeln, dass erwiesenermaßen funktionierte. Um Arbeitsgänge zu vermitteln wurden sog. Job Breakdown Sheets erstellt. Diese zeigten nicht alle Schritte, sondern nur die wesentlichen auf. Auch für die Verbesserung wurden die wesentlichen Schritte (wenn auch etwas anders) notiert. Bei Toyota wurde daraus standardisierte Arbeit und Value Stream Mapping.

Um die Wirksamkeit des Ausbildungsprogramms zu zeigen, mussten die Firmen Zahlen berichten. Welche Zahlen werden wohl während des Krieges wichtig gewesen sein?

- Können die nötigen Mengen geliefert werden?
- Kann der Materialverbrauch gesenkt werden, um mehr Geräte zu liefern?
- Kann die Zeit für die Produktion gesenkt werden?

Vor allem die Prozesseffizienz (Zeit für die reine Bearbeitung in Bezug zur gesamten Zeit von Anfang bis Ende) wurde zu einer wichtigen Kennzahl. Und die Unternehmen lieferten beeindruckende Zahlen. Insgesamt wurden in der Zeit von 1940 bis 1945 1,7 Mio. Menschen in mehr als 16.000 Werken ausgebildet.

[1] Siehe Jeff Sutherland: Origins of Scrum, Blogbeitrag vom 5. Juli 2007, abrufbar unter https://www.scruminc.com/origins-of-scrum/ und

Dieses Wissen gerät nach dem Krieg in den USA in Vergessenheit. Erst die MIT-Studie hat in den 1980er-Jahren dieses Wissen wieder systematisch erfasst (ohne allerdings die Ursprünge zu erwähnen)[2].

In Japan war dieses Wissen sehr willkommen, um die am Boden liegende Wirtschaft wieder aufzubauen. Toyota hat viele Dinge verfeinert. Kaizen bedeutete, täglich viele kleine Dinge zu ändern. Große Sprünge waren während des Krieges nicht möglich. Um mehr zu produzieren, musste man systematisch herausfinden, wie man an anderen Stellen etwas einsparen konnte.

Es entstanden Techniken wie Kanban, um den Bestand an zu lagernden Teilen gering zu halten. David Anderson hat die Prinzipien von Kanban für die Fertigung später für die Aufgabenbearbeitung in einem Wartungsteam genutzt.

Jeff Sutherland und Ken Schwaber greifen in den 1990er Jahren die Ideen von Lean Thinking auf und integriert sie in Scrum. Zusätzlich nutzen sie das Konzept des Chefkonstrukteurs von Toyota, um Produkte inkrementell und mit Feedback zu verbessern.

Jeff Sutherland, Ken Schwaber und 15 weitere Personen kommen im Jahr 2001 zusammen, um ein Agiles Manifest zu erstellen. Der Begriff agil wurde aus dem Buch „Agile competitors and virtual organizations“ übernommen[3].

Bei Lean Thinking, Scrum, Agilität und Kanban ging es nie um das Erfinden von neuen Methoden. Es ging immer darum, bestimmte Ziele zu erreichen und erfolgreiche Produkte schrittweise zu entwickeln.

3 Wertebasis

3.1 Werte im Lean Thinking

Lean Thinking baut auf der Idee auf, Dinge zusammen mit den Personen zu verbessern, die im Betrieb arbeiten. Dazu war ein vernünftiger Umgang miteinander wichtig. Die Grundpfeiler im Lean Thinking sind daher[4]:

- Continuous Improvement (ständige Verbesserung)
- Respect for people (Respekt vor den Menschen)

3.2 Scrum-Werte

Für Ken Schwaber[5] sind die Scrum-Werte das, was einem Scrum-Team Leben gibt. Die Rollen, Artefakte und Ereignisse sind das Skelett. Richtig lebendig wird das Scrum erst, wenn das Scrum-Team die folgenden Werte aktiv lebt und vertrauensvoll zusammenarbeitet:

- Selbstverpflichtung (engl. Commitment)

---

[2] Siehe Womack, James P., et al. Machine that changed the world. Simon and Schuster, 1990.
[3] Siehe Goldman, Steven L., Roger N. Nagel, and Kenneth Preiss. Agile competitors and virtual organizations: strategies for enriching the customer. Vol. 8. New York: Van Nostrand Reinhold, 1995.
[4] Siehe The Toyota Way, https://www.toyota-europe.com/world-of-toyota/this-is-toyota/the-toyota-way
[5] siehe Scrum.org: “Scrum Guide Refresh July 2016 - Scrum Pulse Episode #14“, Youtube-Video, hochgeladen am 13.07.2016, abrufbar unter https://www.youtube.com/watch?v=0hRZffDD1ec

- Fokus (engl. Focus)
- Offenheit (engl. Openness)
- Respekt (engl. Respect)
- Mut (engl. Courage)

Gute agile Teams nehmen sich Zeit, um ihre Wertebasis zu klären. Bei Scrum ist das empirische Arbeiten sehr wichtig. Dazu gehören **Transparenz** und das ständige **Überprüfen** und **Anpassen** von Abläufen und Produkteigenschaften mit zu den wichtigen Dingen.

3.3 Agiles Manifest

Im Februar 2001 haben 17 Erstunterzeichner in einem Ski-Ressort in Utah diese Werte als Agiles Manifest (*engl. Manifesto for Agile Software Development* oder kurz *Agile Manifesto*[6]) formuliert. Der Begriff ergab sich aus dem Buch „Agile Competitors". Agil wurde hier als Lean Thinking mit der Ergänzung verstanden, dass Kunden in den Prozess miteingebunden werden.

Individuen und Interaktionen	sind wichtiger	als Prozesse und Tools
Funktionierende Software	ist wichtiger	als umfangreiche Dokumentation
Kooperation mit Projektbetroffenen	ist wichtiger	als Vertragsverhandlungen
Reaktion auf Änderungen	ist wichtiger	als die Verfolgung eines festgelegten Plans

Es ist nicht so, dass die Dinge auf der rechten Seite nicht gebraucht werden. Die Punkte auf der linken Seite werden aber höher bewertet. Zu diesen Wertepaaren gibt es noch 12 Prinzipien[7]. Nehmen Sie sich Zeit, sich diese Prinzipien anzusehen.

3.4 Auslastung ist kein Maß für Fortschritt oder Erfolg

Viele Organisationen achten darauf, dass ihre Mitarbeiter und Führungskräfte hoch ausgelastet sind. Aber eine hohe Auslastung führt dazu, dass jemand lange warten muss, wenn er die Hilfe oder Zustimmung von anderen braucht.

Gerald Weinberg[8] hat Anfang der 1990er Jahre in Zahlen ausgedrückt (siehe Tabelle 1), wie hoch die Verluste durch das geistige Wechseln (umrüsten) zwischen den Projekten sind.

Anzahl Projekte	Zeitanteil pro Projekt	Verlust durch Umrüsten
1	100%	0%
2	40%	20%
3	20%	40%
4	10%	60%
5	5%	75%

Tabelle 1: Rüstzeiten für das gleichzeitige Bearbeiten von mehreren Aufgaben oder Projekten (Quelle: G. Weinberg)

[6] siehe www.agilemanifesto.org
[7] Siehe www.agilemanifesto.org/principles.html
[8] Siehe Weinberg, Gerald M.: **Quality Software Management**, 1 : Systems Thinking. New York: Dorset House Pub., 1992.

Agile Organisation stellen die Scrum-Teams so zusammen, dass sie konzentriert an einem Projekt arbeiten können, bevor sie das nächste beginnen. Dadurch schafft das gleiche Team mehr Projekte pro Jahr. Eine Auslastung von 100% bedeutet Stillstand.

3.5 Business Value

Viele Organisationen haben ein gutes Kostenbewusstsein. Wenn allerdings das Bewusstsein für den Business Value fehlt, der in der Projektarbeit geschaffen wird, wird bei der Planung eine hohe Auslastung statt einer schnellen Lieferung bevorzugt.

Ein einfaches Modell haben Gane und Sarson[9] bereits Ende der 1970er Jahre angeboten. Die Abkürzung IRACIS weist auf folgende Möglichkeiten hin:

- Mehr Umsatz (*engl. "increased revenue – IR")*
- Vermiedene Kosten (*engl. "avoided cost – AC"*)
- Verbesserte Leistungsfähigkeit (*engl. "improved service – IS"*)

Ein Projekt liefert Business Value, wenn es für Erlöse sorgt oder die Betriebskosten langfristig senkt. Ein Product Owner betrachtet immer Wert und Kosten. Er priorisiert die einzelnen Ideen so, dass möglichst schnell ein guter Wert geliefert wird.

4 Concurrent Engineering und Design Thinking

4.1 Produkte evolutionär entwickeln

Toyota ist mit seiner Produktentwicklung sehr erfolgreich. Diese ist keineswegs besonders, sondern eine sehr alte Methode. Während die meisten Unternehmen in den USA oder Europa diese Methode vergessen haben, hat Toyota sie beibehalten und verbessert. Bis zum Ende des 2. Weltkriegs war sie so weit verbreitet, dass sie keinen besonderen Namen hatte[10]. Heute kennt man diese Techniken unter den Begriffen „Concurrent Engineering", „Design for Manufacture" (DFM) und „Design for Assembly" (DFA).

- Concurrent Engineering: statt die eine perfekte Lösung zu entwickeln, werden mehrere Ideen solange parallel geprüft, bis eine gute Lösung übrig bleibt. Maschinen und Services entwickeln sich evolutionär.
- DFM und DFA sind Techniken, die darauf abzielen, die Komplexität der Produkte zu reduzieren und die Produktion zu vereinfachen.

Die Beschreibung von Nonaka und Takeuchi über die Produktentwicklung[11] in japanischen Firmen hat den Namen Scrum für das erste Scrum Team geprägt.

---

[9] siehe Gane, Chris P., and Trish Sarson. *Structured systems analysis: tools and techniques*. Prentice Hall Professional Technical Reference, 1979.
[10] Ziemke, M. Carl, and Mary S. Spann. "Concurrent engineering's roots in the World War II era." Concurrent Engineering. Springer, Boston, MA, 1993. 24-41.
[11] Siehe Takeuchi, Hirotaka, and Ikujiro Nonaka. "The new new product development game." Harvard business review 64.1 (1986): 137-146.

Allen Ward, Jim Morgan und Jeffrey Liker haben lesenswerte Bücher über die Produktentwicklung bei Toyota geschrieben[12].

4.2 Design Thinking

Design Thinking ist eine Vorgehensweise, die ursprünglich von Mitarbeitern der Design- und Innovationsberatungsfirma IDEO entwickelt wurde, um kreative Lösungen für schwierige Probleme zu finden. Als Entwickler gilt vor allem **David Kelley**, einer der Gründer von IDEO. **Tim Brown**, ebenfalls von IDEO, hat ein wichtiges Buch über Design Thinking („Change by Design“) geschrieben.

Viele Menschen halten sich nicht für besonders kreativ. David Kelley konnte beobachten, wie ein bekannter Psychologe Patienten mit Phobien behandelt hat. Schlüssel zum Erfolg war ein schrittweiser Prozess. Kelley und andere haben mit Design Thinking einen Prozess entwickelt, mit dem JEDER Mensch kreative Lösungen entwickeln kann:

- Verstehe die Menschen, beobachte sie, sammle Daten. Entwickle Empathie für ihre Lage, um das wirkliche Problem zu verstehen.
- Entwickle Idee und baue einfache Prototypen.
- Teste Deine Ideen, sammle Daten und verbessere die Lösungen.

Während die meisten unter Design eine praktische und ästhetische Gestaltung von Gegenständen oder Prozessen verstehen, geht Design Thinking weiter. Weg vom Objekt, hin zur Gesamtlösung.

IDEO hat übrigens eine eigene Seite zu Design Thinking[13] zusammengestellt und bietet ein Leitfaden („*The Field Guide to Human-Centered Design*“) zum Download[14] an.

IDEO unterscheidet drei Phasen (siehe Abbildung 2), wobei es zwischen diesen Phasen auch immer wieder zu Rücksprüngen kommen kann:

- In der **Inspirationsphase** geht es darum, Menschen zu beobachten und ihre Bedürfnisse und Probleme besser zu verstehen.
- In der **Ideenfindungsphase** wird das Verständnis durch schnelle Prototypen vertieft und Lösungen schnell ausprobiert.
- In der **Implementationsphase** wird eine marktreife Lösung entwickelt.

Bei Design Thinking ist es wichtig, dass man verschiedene Sichtweisen im Raum hat, um eine gute Lösung zu finden, die sowohl erstrebenswert als auch technisch machbar und bezahlbar ist.

Im Bereich Design Thinking gibt es eine Vielzahl von Werkzeugen und Ideen. Sehr häufig kommen Customer Journey Maps und Personae zum Einsatz.

---

[12] Siehe Ward, Allen C., and Durward K. Sobek II. **Lean product and process development**. Lean Enterprise Institute, 2014.; Morgan, James M., and Jeffrey K. Liker. **The Toyota product development system**. Vol. 13533. New York: Productivity Press, 2006.
[13] siehe https://designthinking.ideo.com/
[14] siehe http://www.designkit.org

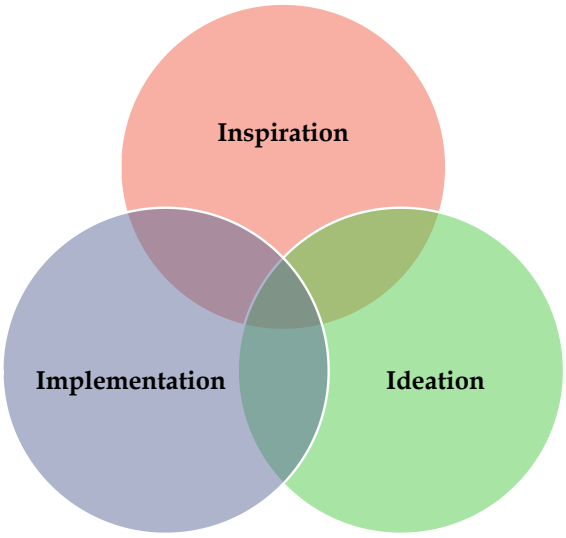

Abbildung 2: Phasen im Design Thinking Ablauf

Es sei darauf hingewiesen, dass jeder Design-Prozess grundsätzlich und ganz klar zwischen Problemformulierung und Lösungsfindung trennt und dass man mit mehr als einem Lösungsvorschlag in die nächste Phase geht. Besonders im IT-Bereich neigen wir dazu, zu schnell mit der erstbesten Lösung zu starten. Diesen Automatismus unterbrechen wir bewusst.

Ein **Customer Journey Map**[15] zeigt die verschiedenen Phasen und Bereiche an, in denen ein Kunde eine Leistung nutzt. Abbildung 3 zeigt die Struktur eines Service Blueprints an, einer Variante vom Journey Map.

Mit solchen Maps soll der Blick von außen auf die Bedürfnisse des Kunden geschärft werden. Tim Brown[16] berichtet von einem Projekt bei Amtrak, in dem das Team die Kunden beobachtet hat. Beim Beschreiben des Customer Journeys wurde klar, dass die Zugbenutzung im Groben aus zehn Schritten besteht (z. B. zum Bahnhof fahren, einen Parkplatz finden, eine Fahrkarte kaufen, den richtigen Bahnsteig finden etc.). Das Einnehmen eines Platzes im Zug war erst Schritt acht. Ein Großteil der Aktivitäten der Kunden hatte also gar nichts mit einem Zug von Amtrak zu tun. Hier konnte das Team viele Ideen ausprobieren, um Zugfahren zu einer angenehmeren Erfahrung zu machen.

[15] siehe Beschreibung bei IDEO: http://www.designkit.org/methods/63
[16] siehe Brown, Tim: Change by Design : How Design Thinking Transforms Organizations and Inspires Innovation. 1. Aufl.. New York: Harper Collins, 2009. S. 94

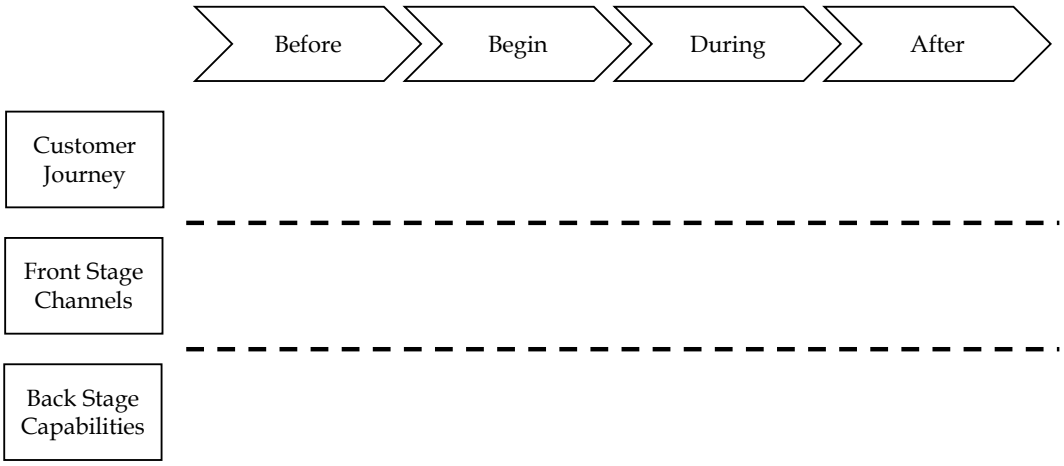

Abbildung 3: Struktur eines Service Blueprints (nach Reason, Lovlie, Flu)

Ein Journey Map – egal ob im Ist- oder im Zielzustand – soll Gespräche im Team und mit Anwendern unterstützen.

User Personas (oder personae) wurden das erste Mal von Alan Cooper[17] als Hilfsmittel zum Verstehen unterschiedlicher Anwender beschrieben. Eine Persona fasst die Eigenschaften eines bestimmten Anwendertyps möglichst konkret und lebendig auf einer Seite zusammen. Neben Name, Alter, Geschlecht könnten ein Foto und weitere Informationen über Ziele, Rahmen oder Hobbys das Bild ergänzen. Unterschiedliche Personas könnten unterschiedliche Customer Journeys beschreiten. Auch hier gilt, dass die Personas das Gespräch fördern.

Bei Design Thinking gibt es die Aufforderung, möglichst schnell zu **Prototypen** zu gelangen, um daran zu lernen, was wirklich wichtig ist. Die Qualität der Prototypen kann sich im Projektverlauf ändern.

5 Empirisches Arbeiten

Ohne Daten wissen wir nicht, ob wir uns verbessern. Ohne Daten können wir keine Annahmen über Verbesserungen treffen.

5.1 Deming-Cycle und Toyota Kata

Das kontrollierte Experimentieren wird auch Deming-Cycle genannt:

- **Plan:** Was haben wir vor? Was erwarten wir?
- **Do:** Welche messbaren oder beobachtbaren Ergebnisse gibt es?
- **Study:** Welche Schlüsse ziehen wir aus den Ergebnissen?
- **Adjust:** Was ändern wir in unseren Abläufen konkret?

[17] siehe Alan Cooper: The origin of personas, Blog von Alan Cooper (Cooper Journal), erschienen am 15. Mai 2008, abrufbar unter http://dev.cooper.com/journal/2016/10/the_origin_of_personas

Toyota hat das ständige Verbessern zur Routine in allen Bereichen gemacht. Mike Rother hat dafür den Begriff **Toyota Kata** geprägt[18]. Der Weg von der aktuellen Situation zum gewünschten Ziel ist oft nicht offensichtlich. Empirisches Denken hilft beim Lernen. Dazu gibt es vier Fragen:

- In welche Richtung wollen wir uns bewegen bzw. vor welcher Herausforderung stehen wir?
- Wie ist die aktuelle Situation konkret?
- Was ist der nächste Zielzustand? (zu einem bestimmten Datum mit messbaren Ergebnissen bei einer bestimmten Arbeitsweise)
- Welche Experimente (mit PDSA) bringen uns näher an den Zielzustand?

Es gibt drei Arten von Experimenten:

- **Go and see, Lernen mit den Füßen:** Man beobachtet (passiv), was eigentlich konkret passiert.
- **Forschungsexperiment:** Man greift bewusst in den aktuellen Prozess ein. Es wird etwas geändert und man sieht sich die Reaktion an.
- **Hypothese testen:** Hier wird auch etwas geändert. Aber das Ziel ist es, eine Hypothese zu bestätigen oder zu widerlegen.

Hilfreich ist ein Versuchsprotokoll, um die Daten und Erkenntnisse zu notieren.

5.2 Durchlaufzeit

Im Bereich Betrieb ist der Begriff „Durchlaufzeit“ zentral. Wenn etwas zu lang dauert, meinen wir, dass die Durchlaufzeit für eine Aufgabe zu lang ist.

Die gesamte Durchlaufzeit von einer Aufgabe setzt sich aus mehreren Zeiten zusammen:

- Rüstzeit
- Bearbeitungszeit
- Liegezeiten (und andere Wartezeiten)
- Transportzeiten

Nur die Bearbeitungszeit ist wertschöpfend. Wenn wir etwas beschleunigen wollen, reicht es nicht, nur die Bearbeitungszeit zu optimieren. Meist ist es viel einfacher Liege- oder Wartezeiten, Transportzeiten und Rüstzeiten zu verkürzen.

Durchlaufzeiten kann man messen und berechnen.

5.3 Prozesseffizienz

Die Prozesseffizienz ist das Verhältnis von reiner Bearbeitungszeit und Kalenderzeit für einen Vorgang.

[18] Mike Rother: The Toyota Kata Practice Guide: Practicing Scientific Thinking Skills for Superior Results in 20 Minutes a Day. Madison: McGraw Hill Professional, 2017.; Mike Rother: Toyota Kata: Managing People for Improvement, Adaptiveness and Superior Results. Madison: McGraw Hill Professional, 2009.

5.4 Warteschlangentheorie

Um Durchlaufzeiten besser zu verstehen, brauchen wir ein Verständnis davon, wie Aufträge in einem System bearbeitet werden. Dies ist Gegenstand der Warteschlangentheorie.

Ihre Gesetzmäßigkeiten gelten sowohl für Kommunikationsnetzwerk und Verkehrssysteme als auch in der Fertigung und Logistik. Sie gelten auch für Aufgaben im Service Management. Abbildung 4 zeigt die Elemente eines Systems mit einer Warteschlange.

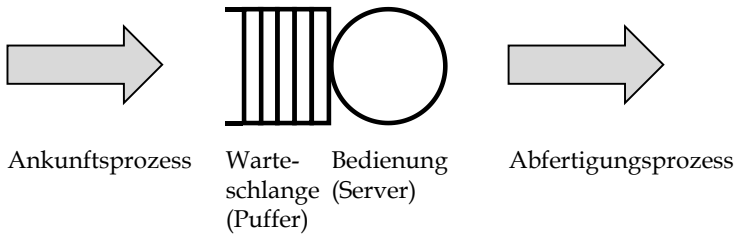

Abbildung 4: Elemente eines Systems mit einer Warteschlange

Bei Warteschlangen gibt es mehrere wichtige Begriffe:

- **Ankunftsprozess:** Die Art und Verteilung, wie neue Kunden (Aufträge, Sendungen, Werkstücke etc.) ankommen.
- **Verteilung der Bedienzeiten:** Wie werden die Aufträge etc. durchschnittlich bedient?
- **Anzahl der Bediener:** Wie viele Einheiten bearbeiten die Kunden?
- **Kapazität der Warteschlange:** Wie viele Aufträgen können warten, bevor sie abgewiesen werden?
- **Population:** Wie viele Aufträge gibt es insgesamt? Meist wird hier eine unendliche Menge angenommen.
- **Abfertigungsprozess:** In welcher Reihenfolge werden ankommende Aufträge bearbeitet.

In Warteschlangen gelten bestimmte Gesetzmäßigkeiten. Das wichtigste ist das **Gesetz von Little**:

- mittlere Anzahl Kunden L = mittlere Ankunftsrate λ x mittlere Aufenthaltsdauer im betrachteten System W
- L = λW *(λ steht für den griechischen Buchstaben Lambda)*

Dazu ein paar Beispiele:

Beispiel 1: pro Woche (40 Arbeitsstunden) kommen 10 Kunden im Abstand von genau 4 Stunden. Die Bearbeitungszeit ist ebenfalls genau 4 Stunden.

- Ankunftsrate λ=10 Kunden/40 Stunden=0,25 Kunden/Stunde
- mittlere Aufenthaltsdauer = 4 Stunden
- mittlere Anzahl Kunden, die warten L = λW = 0,25 Kunden/Stunde x 4 Stunden = 1 Kunde

Beispiel 2: Am Montagmorgen kommen gleichzeitig 10 Kunden (die einzigen in dieser Woche). Die Bearbeitungszeit ist ebenfalls genau 4 Stunden.

- Ankunftsrate λ=10 Kunden/40 Stunden=0,25 Kunden/h
- mittlere Aufenthaltsdauer W $=\frac{4+8+12+16+20+24+28+32+36+40}{10}$ = 22 h
- mittlere Anzahl Kunden, die warten L = λW = 0,25 Kunden/h x 22 Stunden = 5,5 Kunden

Je nachdem, wie das Verhältnis zwischen Ankunftszeit und Bedienzeit ist, ändert sich die Auslastung des Bearbeiters.

- Auslastung = mittlere Ankunftsrate / mittlere Bedienzeit
- $\rho = \frac{\lambda}{\mu}$

In einer typischen Warteschlange im Service Management (Typ M/M/1/∞) errechnet sich die mittlere Anzahl an Aufträgen aus dem Verhältnis von Auslastung durch freie Kapazität:

- Anzahl der Elemente **im** System = $\frac{\rho}{1-\rho}$ *(ρ steht für den griechischen Buchstaben Rho)*
- Anzahl der Elemente in der Warteschlange **vor dem** System = $\frac{\rho^2}{1-\rho}$

Wer zu 50% ausgelastet ist, hat damit im Schnitt eine Aufgabe auf seinem Schreibtisch (0,5/(1-0,5) = 1). Bei 80% Auslastung sind es schon 4 Jobs, bei 95% Auslastung sind es durchschnittlich 19 Jobs.

Das Gegenmittel zu langen Durchlaufzeiten ist das aktive Managen von Warteschlangen.

6 Kanban

6.1 Was ist Kanban[19]?

David Anderson wollte ursprünglich Scrum einführen. Aber das passte nicht zum Aufgabenumfeld. Das Problem lag weniger in der Unsicherheit der einzelnen Aufgaben, sondern darin, dass die Arbeit nicht durch das Team fließen konnte. An verschiedenen Stellen stockte die Arbeit, ohne dass dies dem Team selbst und den anderen Stakeholdern bewusst war.

Anderson hat als erste Maßnahme die Arbeit in einer großen Tabelle auf einer Wandtafel sichtbar gemacht (siehe Abbildung 5). Jede Arbeitsstation entspricht einer Spalte. Jeder Auftrag wird durch eine Karte symbolisiert, die durch die Spalten wandert.

[19] Kanban bedeutet in japanischer Sprache Signalkarte.

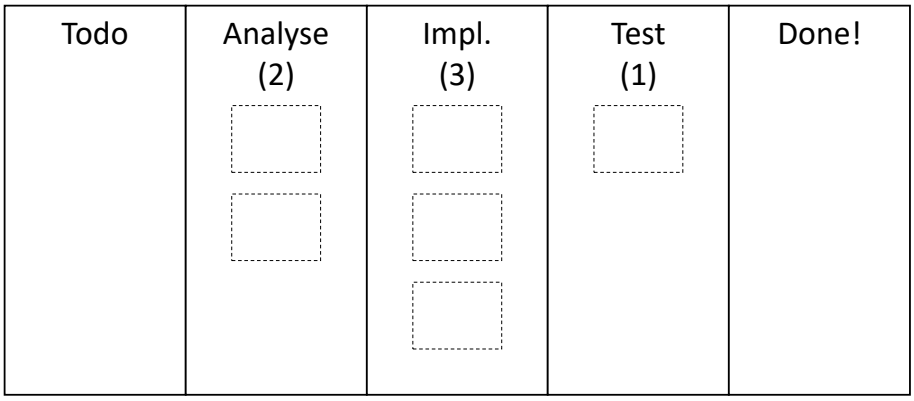

Abbildung 5: Beispiel für ein Kanban-Board

An den einzelnen Arbeitsstationen bilden sich durch die anstehende Arbeit Warteschlangen. Damit sich die Arbeit nicht staut, hat das ursprüngliche Team angefangen, den Zufluss zu begrenzen und so die Länge der Warteschlangen zu begrenzen. Man spricht hier von Limits oder Puffergrößen.

Arbeit darf nur weitergegeben werden, solange ein Platz in der Warteschlange der nächsten Station frei ist. Wenn kein Platz frei ist, muss die Arbeit so lange warten, bis ein Platz frei wird. Die Idee dahinter ist, keine neue Arbeit anzufangen, solange nichts fertig wird.

Damit entsteht ein sog. Pull-System. Statt die Arbeit in das System zu drücken („Push“), wird sie von hinten herausgezogen.

So wurde allen transparent, was zu tun ist. Stück für Stück hat das Team den Kanban-Prozess verbessert. David Anderson hat für Kanban ein Erfolgsrezept definiert:

- Fokussiere auf Qualität
- Reduziere den Work in Progress (WIP)
- Liefere häufig
- Sorge für ein Gleichgewicht zwischen Nachfrage und Durchsatz
- Priorisiere
- Nimm die Quellen von Variabilität ins Visier, um die Vorhersagbarkeit zu verbessern.

Das Ergebnis war, dass Anderson und sein Team, die Durchlaufzeit von Aufgaben dramatisch reduziert haben. Innerhalb von 5 Quartalen wurde sie von 125 auf 25 Tage reduziert[20].

Für Kanban kann man mit einem physikalischen Board starten. Sehr beliebt sind Trello.com und JIRA mit der agilen Ergänzung. Klaus Leopold beschäftigt sich in einem Buch mit immer wieder auftretenden Fragen zu Kanban[21].

[20] Anderson, David J.: Kanban : evolutionäres Change Management für IT-Organisationen. 1. Aufl.. Köln: Dpunkt-Verlag, 2011.
[21] Leopold, Klaus. Kanban in der Praxis: Vom Teamfokus zur Wertschöpfung. Carl Hanser Verlag GmbH Co KG, 2016.

6.2 Serviceklassen

Kanban wird häufig in Betriebsbereichen eingesetzt. Anders als im Entwicklungsbereich ist bei vielen Aufgaben klar, was zu tun ist. Die Frage war also, wann welcher Auftrag wann bedient wird. Dazu hat Anderson Serviceklassen[22] definiert. Beispiele sind:

- **Standard (Standard):** Alle Tickets, die nicht zu den anderen Klassen gehören, zählen zur Standard-Serviceklasse. In der Regel werden Sie nach FIFO behandelt. Das Team kann jedoch auch andere/zusätzliche Regeln definieren.
- **Beschleunigt (Expedite, Speed-Tickets):** Diese Tickets müssen mit hoher Priorität behandelt werden. Je nach Domäne kann es nötig sein, dass das gesamte Team seine momentane Tätigkeit stoppt, um dieses Ticket zu bearbeiten.
- **Fester Termin (Fixed Date):** Wenn eine Funktionalität erst zu einem fixen Termin benötigt wird (zum Beispiel weil dann eine Gesetzesänderung wirksam wird), dann werden die entsprechenden Tickets so durch das Kanban-System geschleust, dass die Funktionalität kurz vor diesem Stichtag produktiv geht.
- **Vage (Intangible):** Wenn der Geschäftswert und/oder die Verzögerungskosten für eine neue Funktionalität vage sind, werden die entsprechenden Tickets nachrangig behandelt. Das Team kann zum Beispiel definieren, dass sich zu jedem Zeitpunkt nur ein solches Ticket im System befinden darf.

Beim Annehmen eines Auftrags wird seine Serviceklasse festgelegt.

6.3 Welche Steuerungsmöglichkeiten hat das Team?

Das Team hat nun die Möglichkeit, mit verschiedenen Elementen des Systems zu spielen, um den Durchsatz zu verbessern:

- Es kann Spalten hinzufügen oder entfernen.
- Es kann die Limits (oder Puffer) in den Arbeitsspalten vergrößern oder verkleinern.
- Es kann die Serviceklassen der Aufgaben verändern.
- Es kann den Mix der Aufgaben verändern.
- Es kann das Board erweitern, also Teammitglieder aus anderen Abteilungen integrieren, um die Gesamtleistung zu verbessern.

Kanban macht hier keine Vorgaben. Es ist die Aufgabe des Teams, regelmäßig nach Verbesserungsmöglichkeiten zu suchen.

6.4 Rollen bei Kanban

Bei Kanban gibt es **keine** speziellen Rollen. Meist findet man folgende Rollen:

- Team: Erledigt die Arbeit
- Teamleiter

[22] siehe http://de.wikipedia.org/wiki/Kanban_%28Softwareentwicklung%29

- (interner oder externe) Kunde: ein oder mehrere Personen, die die Reihenfolge der Aufgaben in der Eingangsspalte festlegen.

6.5 Besprechungen

Bei Kanban gibt es den Willen zur kontinuierlichen Verbesserung (Kaizen). Dazu haben sich in der Praxis folgende Besprechungen etabliert:

- **Nachfüllbesprechung** (Replenishment Meeting): Meeting mit den Kunden darüber, welche Aufgaben in die Eingangswarteschlange kommen.
- **Daily Standup:** Das Team synchronisiert sich in einem kurzen Status-Meeting (max. 15 min). Dazu trifft es sich vor dem Kanban-Board und bespricht vor allem, wie Blockaden aufgelöst werden.
- **Operations Reviews:** Diese monatlichen oder unregelmäßigen Treffen dienen der Verbesserung des gesamten Durchflusses. Dazu nehmen nicht nur das Team sondern auch Führungskräfte und Lieferanten und Abnehmer des Teams teil.
- **Root Cause Analysis:** Diese Meetings dienen dazu, bestimmten Hindernissen oder Problemen auf den Grund zu gehen und sie zu beheben.

6.6 Visuelle Steuerung

Kanban nutzt zur Steuerung zum einen das Kanban-Board und zum anderen CFDs (CFD = Cumulative flow diagram).

Ein CFD wird erstellt, in dem man in regelmäßigen Abständen auf das Kanban-Board sieht und zählt, wie viel Arbeit an den einzelnen Stationen wartet.

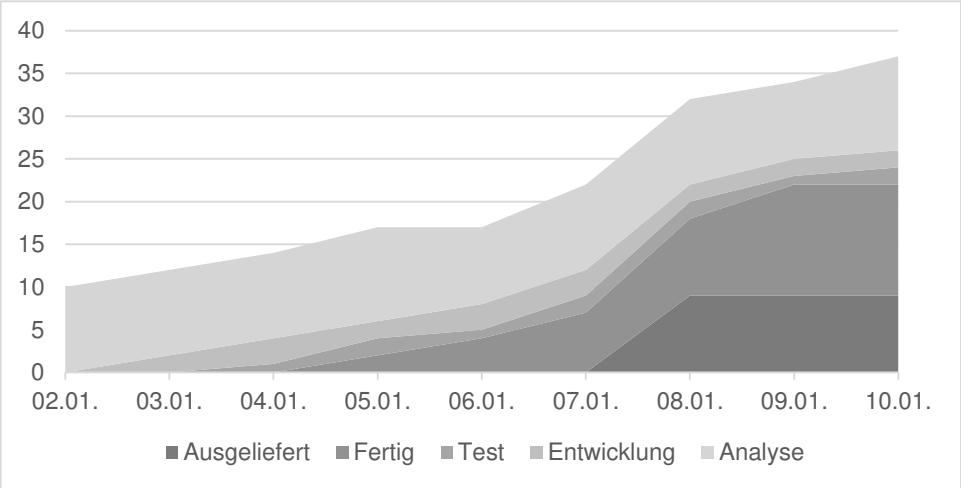

Abbildung 6: Beispiel für ein CFD

Man kann ein CFD auch mit Excel erstellen. Matthias Bohlen und Håkan Forss haben auf ihren Webseiten Anleitungen dazu veröffentlicht[23].

[23] Matthias Bohlen: Cumulative Flow Diagram zeichnen in Excel, abrufbar unter http://mbohlen.de/cumulative-flow-diagram-zeichnen-in-excel/ und Håkan Forss: Cumulative Flow Diagram – How to create one in Excel 2010, Hakan

Mattias Skarin hat im Blog seiner Firma einen Beitrag mit ganz unterschiedlichen Boards und ihren Kontexten veröffentlicht[24].

David Lowe hat ein Video veröffentlicht, dass die Auswirkungen von unterschiedlichen WIP-Limits auf den Durchsatz und die Durchlaufzeiten zeigt[25].

7 Unsicherheit und Komplexität

Um zu verstehen, warum Scrum so populär ist, muss man den Begriff **Unsicherheit** verstehen. Wir nutzen Scrum nicht, weil Scrum modern ist, sondern weil die Unsicherheit gestiegen ist.

Die beiden Wissenschaftler Aaron Shenhar und Dov Dvir haben sehr viel über Kategorisierungsmöglichkeiten von Projekten geschrieben. Das Buch „Reinventing Project Management“[26] ist eine gut lesbare Zusammenfassung ihrer Tätigkeiten. Die Autoren haben zu einem besseren Verständnis von Projekten beigetragen. **Ein Projekt zu führen bedeutet, Ergebnisse unter Unsicherheit liefern.**

Wir kennen viele Formen von Unsicherheit. Interessant ist, dass Shenhar und Dvir von (nur) vier Arten von Unsicherheit ausgehen:

- **Neuartigkeit**: Wie gut kann der Kunde beschreiben, was er will bzw. wie gut kann der Lieferant darstellen, was er liefert?
- **Technologie**: Wie gut kennt sich das Umsetzungsteam mit den Techniken, Methoden, Systemen oder Werkstoffen aus, die es benutzt, um sein Projektproblem zu lösen?
- **Komplexität**: Wie stark hängt die Funktionsfähigkeit des Gesamtergebnisses von der Integration der Einzelleistungen der Umsetzer ab? Wie stark müssen sich die Beteiligten abstimmen, damit das Ganze funktioniert?
- **Zeitdruck**: Wie hoch ist der echte Zeitdruck, d. h. wie hoch ist der Schaden, wenn nicht rechtzeitig reagiert wird?

Zur visuellen Darstellung haben Shenhar und Dvir ein Koordinatensystem mit 4 Achsen gewählt.

Forss's Blog, erschienen am 17.06.2011, abrufbar unter https://hakanforss.wordpress.com/2011/06/17/cumulative-flow-diagram-how-to-create-one-in-excel-2010/

[24] siehe Mattias Skarin: 10 kanban boards and their context updated – v1.5, Blog von crisp.se, erschienen am 04. Juli 2016, abrufbar unter http://blog.crisp.se/2016/07/04/mattiasskarin/10-kanban-boards-and-their-context-updated-v1-5
[25] siehe David Lowe: WIP: why limiting work in progress makes sense (Kanban), Youtube-Video, Veröffentlicht am 07.10.2013, abrufbar unter https://www.youtube.com/watch?v=W92wG-HW8gg
[26] Siehe Shenhar/Dvir 2007: Shenhar, Aaron J., and Dov Dvir. Reinventing Project Management: The Diamond Approach to Successful Growth & Innovation. 1st ed. Mcgraw-Hill Professional, 2007.

Abbildung 7: Ein Koordinatensystem zur Beschreibung von Unsicherheit in Projekten (Quelle: Shenhar, Dvir)

Jede Achse steht für eine Art von Unsicherheit. Jede Achse ist in 3 oder 4 Stufen unterteilt. Stufen nahe an der Mitte stehen für Projekte, die eher einfach zu führen sind. Stufen, die weiter außen sind, deuten auf schwierige Projekte hin.

Jede Achse bestimmt etwas in der Projektsteuerung: der Grad der Neuartigkeit bestimmt, wann die Anforderungen festgelegt werden können, der Grad der technologischen Unsicherheit bestimmt, wann das Design festgelegt wird. Die Anzahl der Beteiligten bestimmt das Maß der Standards und Formalien. Der Zeitdruck bestimmt, wie autonom das Team arbeiten und entscheiden kann.

Stufe	Neuartigkeit	Technologie	Komplexität	Zeitdruck
1	Derivative	Low-Tech	Assembly	Regular
2	Platform	Medium-Tech	System	Fast/ Competitive
3	Breakthrough	High-Tech	Array	Time-Critical
4	-	Super-High Tech	-	Blitz

Tabelle 2: Die Stufen des Rautenmodells

Shenhar und Dvir haben die Tabellen und Abbildungen ihres Buchs kostenlos ins Netz[27] gestellt. Das Rautenmodell von Shenhar und Dvir wurde explizit für Projekte erstellt. Unsicherheit ist gerade für Entwicklungsteams ein Problem. Das Gegenmittel zu Unsicherheit ist Feedback.

Unsicherheit ist ein Merkmal von **komplexen Systemen**. Komplexe Systeme lassen sich nicht so einfach beeinflussen. Dave Snowden hat darüber geforscht und das **Cynefin-Framework**[28] populär gemacht. Das ist ein Modell, um Situationen und das Verhalten von Systemen abzuschätzen. Er definiert fünf unterschiedliche Umgebungsarten:

- Einfache Umgebungen (*engl. simple contexts*): einfache Umgebungen sind stabil und Ursache-Wirkungsbeziehungen sind klar zu erkennen. Beispiel: Kuchen backen.
- Komplizierte Umgebungen (*engl. complicated contexts*): in komplizierten Umgebungen kann es oft mehrere richtige Antworten geben. Es gibt zwar klare Ursache-Wirkungsbeziehungen, aber nicht jeder sieht diese Zusammenhänge. Beispiel: eine Rakete zum Mond schicken
- Komplexe Umgebungen (*engl. complex contexts*): in komplexen Umgebungen gibt es keine allgemein akzeptierte Wahrheit. Das ganze System ist in ständiger Bewegung. Warum Dinge passieren, lässt sich erst im Nachhinein erkennen. Neue Dinge und neue Muster ergeben sich mit der Zeit (Emergenz). In komplexen Systemen gibt es keine Ursache-Wirkungsbeziehungen, nur dispositionelle Eigenschaften. Beispiel: ein Kind großziehen
- Chaotische Umgebungen (*engl. chaotic contexts*): in chaotischen Umgebungen ist es sinnlos nach der richtigen Antwort zu suchen. Weil das System ständig in Veränderung ist, ist es nicht möglich, Wirkungen und ihre Ursachen zu erkennen.
- In der fünften Umgebungsart (*engl. disorder*) herrscht einfach nur Unordnung. Das Schwierige ist, zu erkennen, dass man gerade in dieser Situation ist.

Es gibt mehrere Mitschnitte von den Vorträgen von Dave Snowden, die bei YouTube veröffentlicht wurden.

8 Scrum

8.1 Überblick

Scrum wurde entwickelt, um in unsicheren Situationen schnell Ergebnisse liefern zu können. Mit Scrum wird Feedback organisiert. Anhand des Feedbacks überlegen die Beteiligten, wie und wo sie nachsteuern, um das Ziel zu erreichen.

Scrum[29] definiert dazu nur drei Rollen, weil die Umsetzung schwieriger wird, je mehr unterschiedliche Rollen beteiligt sind:

- Development Team oder kurz Team: diejenigen, die die Arbeit machen
- Product Owner: maximiert den Wert des Ergebnisses und der Arbeit des Teams

[27] siehe Harvard Business Review Press Books in the Classroom, specific course materials, abrufbar unter http://hbsp.harvard.edu/list/syllabi und http://reinventingprojectmanagement.com/050_Teaching.html
[28] siehe Snowden 2007: David J. Snowden & Mary E. Boone: A Leader's Framework for Decision Making. In: Harvard Business Review. November 2007, S. 69–76
[29] Scrum ist keine Abkürzung, sondern bedeutet „Gedränge" in der englischen Sprache.

- Scrum Master: achtet auf den Prozess und die kontinuierliche Verbesserung

Alle drei Rollen zusammen werden als Scrum-Team bezeichnet. Alle anderen Rollen werden pauschal als Stakeholder bezeichnet.

Der Scrum Guide schlägt folgende Artefakte zur Steuerung vor:

- Product Backlog
- Sprint Backlog
- Product Increment

Mit diesen Artefakten soll die Zusammenarbeit erleichtert werden. Sie können bei Bedarf durch weitere Dokumente ergänzt werden.

Bei Scrum arbeiten wir in kurzen Zyklen, den Sprints, damit wir uns immer wieder Feedback holen und verbessern. Zur Koordination kennt Scrum folgende Ereignisse:

- Sprint
- Sprint Planning
- Daily Scrum
- Sprint Review
- Sprint Retrospective

Dazu gibt es das Backlog Refinement als laufende Aktivität, um aus groben Ideen konkrete Ideen zu machen. Die offiziellen Regeln zu Scrum sind im Scrum Guide zusammengefasst. Diesen können Sie sich kostenlos von der Webseite scrumguides.org herunterladen.

8.2 Rollen in Scrum

Scrum definiert aus verschiedenen Gründen bestimmte Rollen.

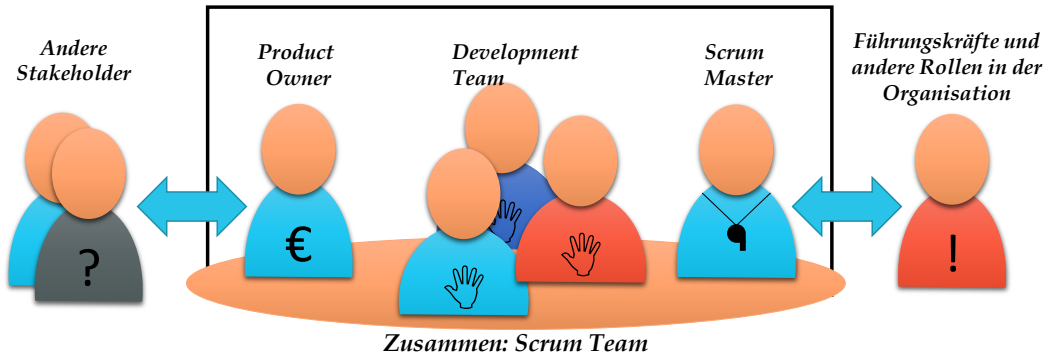

Abbildung 8: Ein Scrum-Team besteht aus Product Owner, Development Team und Scrum Master.

Wir brauchen einen Vertreter für die Business-Seite, den **Product Owner**:

- Er ist der Mini-CEO für das Projekt. Ist verantwortlich für das finanzielle Ergebnis des Projekts (ROI).

- Er hat die Hoheit über das Product Backlog und entscheidet über Anforderungen und definiert Qualität.
- Nur er wählt die nächsten Anforderungen für das Feedback aus und priorisiert Anforderungen.
- Nimmt Features im Review ab.
- Bestimmt nicht, WIE das Team die Arbeit erledigt.
- Greift nicht in die Selbstorganisation des Teams ein.

Der Product Owner ist der Wertmaximierer (Wert der Arbeit des Teams, Wert des Produkts, Wert für die Stakeholder). Der PO kann sich bei der Priorisierung vom Team beraten lassen. Jeff Sutherland berichtet aus seinem ersten Scrum-Team: dem Entwicklungsteam lagen zu viele Ideen vor, um sie alle umzusetzen. Aus diesem Grund wurde eine Person abgestellt, die nichts anderes zu tun hatte, als die besten Ideen herauszufiltern und das Product Backlog zu sortieren. Jeff Sutherland wurde dabei von der Rolle des Chefkonstrukteurs[30] (engl. **Chief Engineer**) bei Toyota inspiriert.

Das **Entwicklungsteam** muss alle Fähigkeiten haben, um Arbeit (aus Kundensicht) abschließen zu können. In einem Softwareteam sind deshalb nicht nur Programmierer, sondern auch Tester und Designer. Solche Teams können schneller Ergebnisse liefern und damit früher Feedback einholen. Deswegen ziehen agile Firmen sog. **Feature Teams** den technischen oder den sog. Komponententeams vor.

- Funktionsübergreifend (*Cross-Funktional*)
- 3-9 Personen
- Bestimmt, **wie** die Arbeit umgesetzt wird
- Vollzeitmitglieder
- Schätzt Entwicklungsaufwände
- Verpflichtet sich selbst seinen Zielen, bestimmt die Arbeitsmenge pro Sprint

Spezialisten fangen an ihr Wissen zu teilen. Sie packen mit an, auch wenn sie ihren angestammten Wissensbereich dafür verlassen müssen. Um Wissenslücken im Team sichtbar zu machen, kann das Team für sich eine Skillmatrix erstellen (siehe Abbildung 9).

[30] Mehr zur Rolle beim Lean Enterprise Institute: https://www.lean.org/shook/DisplayObject.cfm?o=906

	Anton	Bernd	Carola	Dani
Kompetenz 1	7	7	8	6
Kompetenz 2	7	5	5	1
Kompetenz 3	3	3	2	7
Kompetenz 4	6	7	8	7
Kompetenz 5	2	6	4	1
Kompetenz 6	4	6	4	8
Kompetenz 7	3	8	3	5

Abbildung 9: Beispiel für eine Skillmatrix eines Teams (rot 1-3, gelb 4-6, grün 7-9)

Der **Scrum Master** ist Moderator und Coach und hilft so dem Team und PO.

- Ist verantwortlich für die Einhaltung von Scrum-Werten und –Techniken.
- Beseitigt Hindernisse
- Stellt sicher, dass das Team produktiv arbeiten kann
- Unterstützt die Zusammenarbeit zwischen allen Rollen
- Vertritt Scrum gegenüber dem Management
- Schützt das Team vor äußeren Störungen

Die Scrum Master sind die internen Change Agents für den Wandel in Richtung Agilität im Unternehmen, denn sie verstehen am besten, was Agilität und Selbstorganisation bedeuten. Der Scrum Master ist keine Scrum-Polizei.

8.3 Ereignisse in Scrum

Der Scrum Guide empfiehlt, in einer kurzen Taktung, in sog. Sprints, zu arbeiten. Jeder Sprint dient dazu, einen Zwischenstand – ein Inkrement – zu liefern. Ein Sprint ist nicht mit einer Projektphase zu verwechseln. Eine Taktung hilft dabei, sich regelmäßig neu zu orientieren.

Jeder Sprint beginnt mit einem Sprint Planning und endet mit Sprint Review und Sprint Retrospektive. Jeden Tag überprüft das Entwicklungsteam im Daily Scrum, was es in den nächsten 24 Stunden tun kann, um das vereinbarte Sprint-Ziel zu erreichen. Der Scrum Guide empfiehlt für jedes dieser Ereignisse eine Maximallänge, eine Timebox.

Die Sprints folgen ohne Pause aufeinander. Auf Review und Retro des einen Sprints folgt sofort das Planning des nächsten.

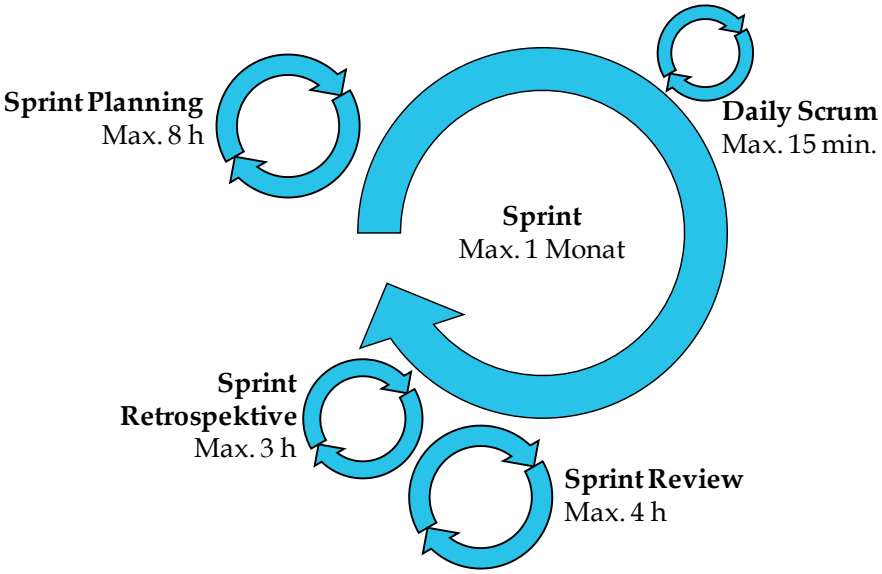

Abbildung 10: Die Scrum Ereignisse im Überblick

Die Ergebnisse des **Sprint Plannings** sind:

- ein konkreter Arbeitsplan für das Entwicklungsteam, der darlegt, wie es das Sprint-Ziel erreichen will,
- eine Prognose darüber, wie viel Arbeit das Entwicklungsteam im Sprint fertigstellt sowie
- ein motivierendes Sprint-Ziel, das dem Entwicklungsteam hilft, sich im Sprint zu fokussieren.

Dieser Arbeitsplan gehört dem Team und wird Sprint Backlog genannt.

Im **Daily Scrum** prüft das Entwicklungsteam, ob es das vereinbarte Sprint-Ziel erreicht. Es plant im Daily Scrum die nächsten 24 Stunden. Jeder im Entwicklungsteam erfährt, was die anderen machen und ob man sich gegenseitig helfen kann. Es ist also kein Statusbericht der einzelnen Mitglieder des Entwicklungsteams an den Scrum Master oder den Product Owner.

Für die Fortschrittskontrolle nutzen Scrum Teams oft sog. Burndown Charts. Diese zeigen an, wie viel Arbeit (gemessen in relativem Aufwand, z. B. in der Einheit Story Points) noch offen ist, bevor der Sprint endet.

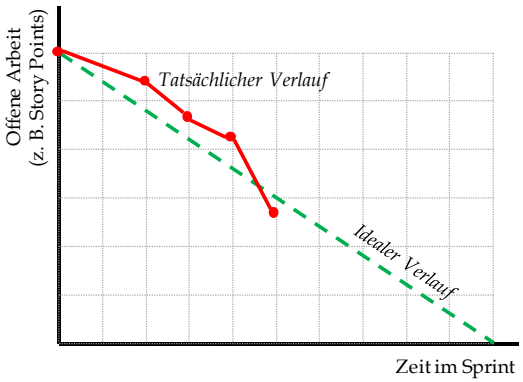

Abbildung 11: ein Beispiel für ein Burndown Chart

Der **Sprint Review** bietet dem Scrum-Team und Stakeholdern die Gelegenheit, an Hand des Inkrements den aktuellen Stand zu bewerten und über die nächsten Entwicklungsschritte zu entscheiden. Er ist keine Abnahme und kein Statusmeeting. Alle Beteiligten nutzen dieses Ereignis, um den aktuellen Stand zu begutachten. Der Product Owner kommt schon mit konkreten Ideen ins Review, die er im nächsten Sprint umsetzen möchte. Im Review überprüft er diese Ideen.

Die **Sprint Retrospektive** ist ein bewusster Haltepunkt, damit sich das Scrum-Team Zeit für Verbesserungen nimmt. Ein häufiger Ablauf folgt einem Vorschlag von Esther Derby und Diana Larson durchgesetzt[31]:

- Türöffner (*engl. Set the stage*)
- Daten sammeln (*engl. Gather data*)
- Einsichten generieren (*engl. Generate insights*)
- Entscheidungen treffen (*engl. Decide what to do*)
- Abschluss (*engl. Close the retrospective*)

Im Netz gibt es den Retromat[32], eine Webseite, die per Zufallsgenerator Abläufe für eine Retrospektive vorschlägt. Marc Löffler[33] und Ralf Dräther[34] haben zwei Anleitungen in deutscher Sprache erstellt.

Es ist das Ziel einer Sprint Retrospektive, die **Lieferfähigkeit** im nächsten Sprint zu verbessern. Das Scrum-Team konzentriert sich auf 1-2 Verbesserungsmaßnahmen, die es im nächsten Sprint konkret umsetzt und dann prüft, ob sich die Lieferfähigkeit verbessert.

8.4 Artefakte in Scrum

Der Scrum Guide empfiehlt drei sog. Artefakte, mit deren Hilfe die Zusammenarbeit organisiert wird. Das Product Backlog mit den Product Backlog Items (PBIs) stellt den aktuellen Stand der Ideen dar, die dem

---

[31] Siehe Derby, Esther ; Larsen, Diana: **Agile Retrospectives** : Making Good Teams Great. 1. Aufl.. Dallas, Texas: Pragmatic Programmers, LLC, 2006
[32] Siehe http://www.plans-for-retrospectives.com
[33] siehe Marc Löffler: Retrospektiven in der Praxis: Veränderungsprozesse in IT-Unternehmen effektiv begleiten, Auflage: 1., Heidelberg, dpunkt.verlag GmbH, 2014
[34] siehe Dräther, Rolf: Retrospektiven – kurz & gut. Köln: O'Reilly Germany, 2014.

Produkt Wert hinzufügen. Das Sprint Backlog zeigt den Arbeitsplan für einen Sprint. Mit Hilfe des Inkrements stellen wir fest, wo wir wirklich stehen.

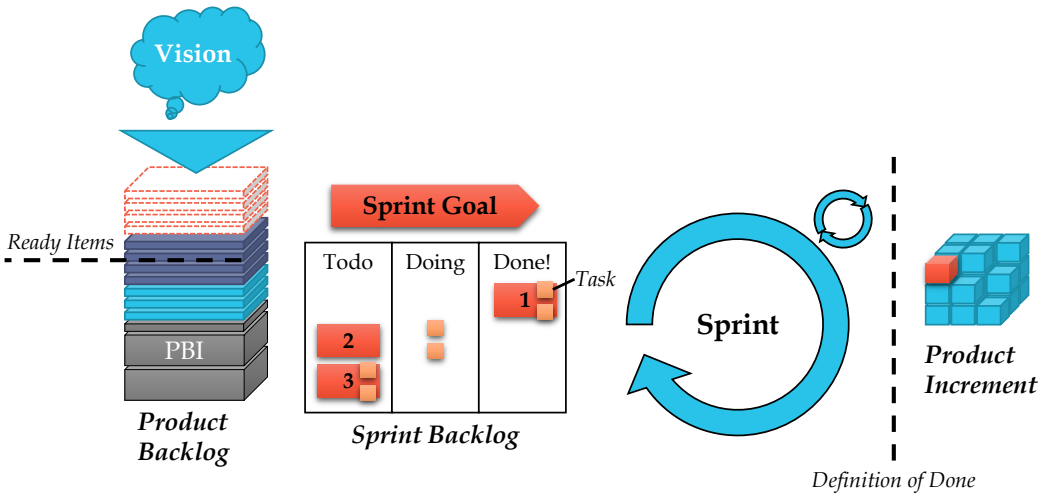

Abbildung 12: Die Scrum Artefakte im Überblick

Das **Product Backlog** ist eine offene, sortierte Liste mit den nächsten Ideen, die das Ergebnis enthalten soll. Die wichtigsten Ideen stehen immer oben. Das Product Backlog ist keine Todo-Liste. Es enthält Teilergebnisse oder Teilziele, die im Ergebnis umgesetzt werden. Das Product Backlog wächst und verändert sich ständig, weil das Scrum-Team kontinuierlich dazu lernt.

Der Scrum Guide bevorzugt kein bestimmtes Format. Er spricht pauschal von Product-Backlog-Einträgen (oder *engl. product backlog items*, kurz PBI). Gut vorbereitete PBIs enthalten **Akzeptanzkriterien**. Jeder PBI hat eigene Akzeptanzkriterien, die dem Entwicklungsteam Hinweise geben, wie es prüfen kann, ob es den PBI erfolgreich umgesetzt hat. Manche PBIs sind im Format **User Story** geschrieben: Als <Rolle> tue ich <eine Aktion>, um <ein Ziel> zu erreichen.

Das Product Backlog wird übrigens kontinuierlich überarbeitet und verfeinert. Dazu reserviert das Scrum Team Zeit im laufenden Sprint (**Backlog Refinement**). Zum Refinement gehört auch das Schätzen des Aufwands (z. B. relativer Aufwand in Story Points mit Planning Poker).

Das **Sprint Backlog** enthält alle Product Backlog Einträge, die für einen Sprint ausgewählt wurden. Dazu kommen die Einzelaufgaben und sonstige Dinge, die beschreiben, wie ein Entwicklungsteam die Backlog Items umsetzt. Es enthält auch mindestens eine Maßnahme zur Prozessverbesserung.

Pro Sprint vereinbart das Scrum-Team ein Sprint-Ziel, damit alle Beteiligten fokussiert am Ergebnis arbeiten.

Das **Inkrement** ist das Ergebnis eines Sprints. Genau genommen ist es das Ergebnis des aktuellen Sprints zuzüglich der Ergebnisse aus den vorherigen Sprints. Das Inkrement ergänzt die Funktionalität aus den vorangegangen Sprints. Das Inkrement dient dazu, damit sich alle Beteiligten Feedback holen können:

- Haben wir Anforderungen und Bedürfnisse der Anwender richtig verstanden?

- Beherrschen wir die Technologie, mit der wir arbeiten?
- Funktioniert das Design?
- Was haben wir beim Erstellen des Inkrements gelernt?

Der Zweck eines Sprints ist immer, ein Inkrement zu liefern. Ohne Inkrement können wir nicht abschätzen, wo wir wirklich stehen. Ob der aktuelle Stand ausgeliefert wird, entscheidet der Product Owner.

Alle Product Backlog Items, die fertig sind, werden ins Inkrement integriert. Daher brauchen wir ein gemeinsames Verständnis davon, was fertig *(engl. done*) bedeutet. Das Scrum-Team vereinbart dazu eine sog. „Definition of Done" (kurz DoD).

9 Change Management und Agilität

Agilität bedeutet, schnell Feedback zu holen, um zu prüfen, ob man auf dem richtigen Weg ist. Über kurz oder lang stellt sich für die Betriebseinheiten die Frage, wie schnell man Änderungen in den Betrieb übernehmen kann.

Leistungsfähige Rechenzentren haben ihr Change Management im Griff, weil sie die Erfahrung gemacht haben, dass in vier von fünf Fällen der letzte Change die Ursache für eine Störung im System war. Tabelle 3 zeigt das Verbesserungspotenzial in der Zusammenfassung[35].

IT-Org	Anzahl von Changes	Fehlgeschlagene Changes	Reparaturzeit (MTTR)	Anteil ungeplante Arbeit
High Performer	> 1000 CRs/Woche	< 1%	Minuten	< 5%
Durchschnitt	Unbekannt bis hunderte/ Woche	Ca. 30-50%	Stunden, Tage	35-45 %

Tabelle 3: Anteil an ungeplanter Arbeitszeit als Indikator für effektives Change-Management (Quelle GTAG2, S. 18)

Die Fehlersuche in kleinen Änderungen ist mit weniger Aufwand verbunden als die Suche in großen Änderungspaketen.

9.1 VisibleOps

Gene Kim, George Spafford und Kevin Behr haben ein paar Jahre lange verschiedene IT-Abteilungen beobachtet. Als gute IT-Abteilungen den Zusammenhang zwischen Changes und Störungen herausgefunden hatten, war klar, was die beste Reaktion auf Systemausfälle ist: Nicht lange analysieren, sondern erst den letzten Change zurücknehmen. Das System läuft wieder und die IT hat Zeit, sich den Fehler genauer anzusehen.

Damit haben die IT-Abteilungen Zeit gewonnen und Stress reduziert. Diese Zeit haben sie in das Automatisieren gesteckt. Sie haben mit den Werkzeugen, die sie hatten, Skripte erstellt, die Changes automatisch verteilen aber auch wieder schnell zurücknehmen. Das hatte einen weiteren Vorteil: Die

[35] siehe Global Technology Audit Guide (GTAG) 2: Change and Patch Management Controls: Critical for Organizational Success, http://www.theiia.org/guidance/technology/gtag2/

Spezialisten, die ja selten Zeit haben, brauchen sich nicht mehr um Standardänderungen zu kümmern. Sie nehmen sich nun die wirklich schwierigen Fälle vor.

Es gab noch eine weitere Verbesserung: Das Team hatte nun Zeit, über anstehende Änderungen zu sprechen. Change Management bedeutet nicht, jeden Change abzublocken. Sondern es geht darum, gemeinsam zu überlegen, ob an alles gedacht wurde, was schief gehen kann, damit man besser vorbereitet ist. Standardchanges wurden automatisch freigegeben. Wieder Zeit gewonnen.

Das Visible Ops Handbook[36] beschreibt diese Schritte gut und verständlich. Beim Lesen wird sofort klar, dass ein Umsetzungsschritt den nächsten zeitlich finanziert ("self-fueling steps"). Die Hauptschritte sind:

- Change-Prozess „ausrufen“, CAB, electrify fences
- Sukzessive Infrastruktur nach Ausfallquellen durchforsten
- Standardchanges definieren, Changes automatisieren (Roll-out UND Rollback)
- Kontinuierliche Verbesserungen planen

Aber wie groß ist die Ersparnis? Gene hat auf der Velocity-Konferenz im Juni 2011 die Unterschiede aufgezeigt[37]. Im Vergleich zum normalen (chaotischen) Vorgehen können IT-Abteilungen mit Visible Ops mehr erreichen:

- 6mal mehr Systeme betreuen
- 8fache Anzahl an Projekten und IT-Services
- der Anteil der ungeplanten Arbeitszeit geht von 35-40 % auf unter 10 % zurück
- 10mal kürzere Reparaturzeit bei kritischen Systemausfällen

Große Ersparnis bei relativ wenig Aufwand. Zudem muss man keine teuren Tools kaufen.

9.2 Continuous Delivery

Continuous Delivery (CD) bezeichnet eine Sammlung von Techniken, Prozessen und Werkzeugen, die den Softwarelieferprozess verbessern. Techniken wie Testautomatisierung, kontinuierliche Integration (Continuous Integration) und kontinuierliche Installation erlauben die Entwicklung qualitativ hochwertiger Software, die durch automatisierte Release-Erstellung auf Entwicklungs-, Test-, Integrations- und Produktivumgebung eingespielt werden kann. Die Automatisierung der Test- und Lieferprozesse ermöglicht es, schnell, zuverlässig und wiederholbar zu liefern und Erweiterungen und Fehlerkorrekturen mit geringem Risiko und niedrigem manuellem Aufwand in die Produktivumgebung oder zum Kunden zu bringen.

[36] Kim, Gene ; Spafford, George ; Behr, Kevin: The Visible Ops Handbook : Implementing ITIL in 4 Practical and Auditable Steps. Revised First Edition. Eugene, Oregon: Information Technology Process Institute, 2005, bestellbar unter http://www.itpi.org/the-visible-ops-handbook-review.html

[37] Siehe Gene Kim: Creating the Dev/Test/PM/Ops Supertribe: From "Visible Ops" To DevOps, Vortrag auf der Velocity 2011 Konferenz, gehalten am 15.06.2011, Vortragsfolien abrufbar unter http://velocityconf.com/velocity2011/public/schedule/detail/21123

Die Autoren[38] geben dazu folgende Prinzipien an:

- Baue einen zuverlässigen, wiederholbaren Auslieferungsprozess auf.
- Automatisiere fast alles
- Stelle alles unter Versionskontrolle
- Wenn es schmerzt, release noch häufiger und ziehe den Schmerz nach vorn.
- Baue Qualität ein
- Fertig bedeutet released
- Jeder ist für den Releaseprozess verantwortlich.
- Kontinuierliche Verbesserung

Letztendlich steckt dahinter die Idee, Infrastruktur genauso wie Programmcode zu behandeln. Um neue Software schnell zu veröffentlichen wird eine sog. Deployment Pipeline aufgebaut.

Bei CD hat sich eine Reihe von Techniken und Tools etabliert.

9.3 Dev2Ops bzw. DevOps

DevOps beschreibt Maßnahmen, um häufige Bruchstellen zwischen Anwendungsentwicklung (Development) und IT-Betrieb (Operations) in Unternehmen zu überwinden.

Gene Kim et al. schlagen im DevOps Handbook[39] drei Wege vor:

- 1. Weg: Flow erhöhen (vor allem durch konsequente Automatisierung)
- 2. Weg: Feedback erhöhen
- 3. Weg: Kontinuierlich lernen

10 Open Space und andere Großgruppenmethoden

Bei bestimmten Gelegenheiten ist es einfacher, alle, die von Änderungen betroffen sind, in den Veränderungs- oder Entscheidungsprozess miteinzubinden. Beispiele: große organisatorische Änderungen, Aufteilung in Scrum-Teams in großen Projekten.

Dazu gibt es ein verschiedene Moderationsmethoden für große Gruppen:

- Open Space Technology (von Harrison Owen)
- Die Zukunftskonferenz (**Future Search**) ist ein mindestens zweitägiges Ereignis (mit zwei Übernachtungen), das einem festen Format folgt.
- Wertschätzende Organisationsentwicklung (**Appreciative Inquiry** - AI) ist ein Ansatz, eine Organisation aus ihren Stärken heraus weiter zu entwickeln.

[38] Siehe Humble, Jez ; Farley, David: Continuous Delivery : Reliable Software Releases through Build, Test, and Deployment Automation. 1. Aufl.. Amsterdam: Pearson Education, 2010.
[39] siehe Kim, Gene ; Debois, Patrick ; Willis, John ; Humble, Jez: The DevOps Handbook : How to Create World-Class Agility, Reliability, and Security in Technology Organizations. United States: IT Revolution, 2016.

- **World Café** pflegt die Kunst des Dialogs. Durch eine gut gestellte Frage erarbeiten sich die Teilnehmer an Tischen ein Thema intensiver und tauschen sich aus.

Open Space ist sehr einfach und wird häufig in agilen Gruppen genutzt. Die Gruppe bestimmt selbst die Themen, über die sie sprechen möchte. Dazu kann am Anfang ein Motto festgelegt werden. Bei Open Space gibt es vier Prinzipien und ein Gesetz.

Bei Open Space spielt die Einladung eine wichtige Rolle.

Open Space besteht aus drei Teilen:

- Zu Beginn wird Open Space erklärt. Danach werden die Teilnehmer aufgefordert, ein Thema zu nennen, für das Sie Leidenschaft entwickelt haben. Einer nach dem anderen kommt vor die Gruppe und nennt sein Thema. Alle Themen werden auf einer Art Stundenplan (der Anliegenwand) notiert.
- Danach beginnt die Marktplatzphase: Alle Teilnehmer tragen sich mit ihrem Namen oder Kürzel bei den Themen ein, die sie interessieren. Wenn jemand zwei Sessions besuchen will, die gleichzeitig stattfinden, verhandelt er mit den Diskussionsleitern über Änderungen am Stundenplan.
- Dann beginnen die eigentlichen Open Space Sessions. Dafür stellt der Organisator verschiedene Räume bereit. Dazu gibt es zum Beispiel einen Stundentakt, nach dem gewechselt wird.

Der Erfinder von Open Space hat ein gutes Buch[40] als Anleitung geschrieben, das wir sehr empfehlen. Für die Community gibt es die Plattform http://www.openspaceworld.org/.

Open Space Technology
Prinzipien:
- Wer auch immer kommt, es sind die richtigen Leute
- Was auch immer geschieht, es ist das Einzige, was geschehen konnte
- Es beginnt, wenn die Zeit reif ist
- Vorbei ist vorbei – Nicht vorbei ist Nicht-vorbei

Gesetz:
- Das Gesetz der zwei Füße

Lass dich darauf ein, überrascht zu werden.

11 Abschluss

Nun haben Sie einen Einblick in die Vielfalt der Agilität bekommen. Vielleicht können Sie die verschiedenen Begriffe besser einordnen. Denken Sie immer daran, dass es niemals um die Einführung neuer Methoden geht. Wir nutzen eine wissenschaftliche Arbeitsweise, um Abläufe und Produkte zu verbessern.

Unterschätzen Sie nicht die Wirkmechanismen in komplexen Systemen. Oft können solche Systeme kleine Veränderungen noch gut abpuffern. Aber irgendwann verändert sich das ganze System mit einem Schlag

[40] siehe Owen, Harrison: Open Space Technology : A User's Guide. San Francisco: Berrett-Koehler Publishers, 2008.

(*punctuated equilibrium*). Nehmen Sie sich also Zeit, um sich mit den Konzepten in diesem Skript vertraut zu machen. Bleiben Sie dran. Gehen Sie kleine Schritte.

Der Scrum Guide fasst in der Einleitung die Erfahrungen mit Scrum zusammen:

„leicht zu lernen aber schwer zu meistern."

Tauschen Sie sich deshalb mit anderen Interessierten im eigenen Unternehmen, in der Branche oder in der Region aus. Beobachten Sie, was sich in der Community tut.

In verschiedenen Regionen finden regelmäßig Konferenzen statt, die dem Austausch in der Community dienen. Dort besteht oft die Gelegenheit, engagierte Mitglieder der Community zu treffen.

In vielen deutschen Städten gibt es auch regelmäßige Treffen. Die Namen sind unterschiedlich: ScrumTisch, Agile User Group, Agile Meetup, Agile Monday, Agile Wednesday, Lean Coffee. Diese vernetzen sich meist via Xing, LinkedIn oder Meetup. Die Teilnehme kostet nichts. Meist bitten die Organisatoren aber um eine Anmeldung.

Die Trainer*innen und Autor*innen dieses Skripts freuen sich, von Ihnen zu hören. Teilen Sie Ihre Erfahrungen mit und uns und helfen Sie anderen, sich zu verbessern.