# Gegenüberstellung der agilen Prozesse

Vor der Durchführung der drei Projekte haben die Teams eine Gegenüberstellung der agilen Prozesse durchgeführt. Die Ergebnisse dieses theoretischen Vergleichs wurden in Kapitel (X) vorgestellt. Während und nach der Durchführung wurden die vorher aufgestellten Vergleichskriterien in der Praxis umgesetzt und positiv oder negativ bewertet. Eine Zusammenfassung dieser Auswertung findet sich in diesem Kapitel.

## Iterationen

Die Iterationslänge wurde bei Scrum und Crystal Clear gleichermaßen auf eine Woche festgelegt. Für beide Prozesse wurde die Länge als passend für die Planung und Übersicht über den Fortschritt des Teams bei der Entwicklung empfunden. Bei Kanban veränderte sich die Sprintlänge während des Projekts, so wie es die Methode erlaubt. Sie schwankte zwischen einer und zweieinhalb Wochen. Auch hier entschied das Team sich jedoch zum Ende des Projekts dafür zur einwöchigen Iteration zurückzukehren, da es den Entwicklern während der längeren Phasen schwer fiel, den Überblick über die Planung und den Entwicklungsstand zu behalten. Dies liegt jedoch insbesondere an der mangelnden Erfahrung der Teammitglieder beim Schätzen. Besonders schwierig wurde es, nachdem die Iterations- und Release-Zyklen nicht mehr synchron liefen. Auch dies erfordert einiges an Erfahrung der Teammitglieder in der Organisation von Projekten. Die große Flexibilität von Kanbans Iterationen kann demnach von einem unerfahrenen Team vermutlich nicht sofort genutzt werden.

## Teamgröße und Rollen

Das eigentliche Entwickler-Team bestand bei allen Prozessen nur aus zwei Personen. Die dritte Person nahm jeweils eine andere Rolle im Prozess ein. Da es bei Kanban eigentlich keine explizit festgelegten Rollen gibt, gab es lediglich einen Kunden und einen Board-Verantwortlichen. Die geringe Größe des Entwicklungsteams machte die Synchronisation sehr einfach und verursachte kaum Overhead bei der Organisation. Allerdings entstehen vermutlich bei mehr Mitarbeitern auch mehr Verbesserungs- und Veränderungsvorschläge. Außerdem fehlte beiden Entwicklern jemand, der ab und zu die Leitung oder Organisation des Teams übernimmt. Vermutlich fällt es allgemein schwer, von einer Organisationsform mit Projektleiter oder Ähnlichem zu einer so freien Form der Selbstorganisation zu wechseln.

Für ein Scrum-Team ist die Anzahl zu gering. Scrum Master und Product Owner / Kunde mussten in einer Person existieren. Dies ist denkbar schlecht, da der Scrum Master die Aufgabe hat, das Team vor Eingriffen des Kunden während des Sprints zu schützen – in diesem Fall also gegen sich selbst. Beide Rollen sollten unbedingt getrennt gehalten werden. Die Menge der Entwickler war dagegen kein Problem. Allerdings würden bei einem größeren Projekt auch mehr Entwickler nötig sein, um das Vorankommen des Projekts zu gewährleisten.

Ähnliches gilt für Crystal Clear. Auch hier war die Team-Größe von zwei Entwicklern für die Größe des Projekts angemessen und gut zu koordinieren. Die Rollen des Anwenders und Auftraggebers konnten sehr gut in einer Person vereinigt werden. Manchmal fehlte dem Team jedoch eine Art Koordinator – eine Rolle, die bei genügend Mitarbeitern dem Chefdesigner bei der Organisation hilft und ihn freier für die Entwicklung selbst gemacht hätte.

## Techniken

Weder Scrum noch Kanban schreiben das Verwenden bestimmter Techniken vor. Deshalb wurde aus dem Wissensrepertoire der Entwickler Pair Programming gewählt und in beiden Projekten regelmäßig umgesetzt. Alle Entwicklerteams empfanden dies als sehr hilfreiches Mittel zur Wissensverteilung und –erweiterung und würden es jederzeit wieder anwenden, auch wenn damit zuerst die doppelte Zeit für eine Aufgabe aufgewendet wird. Dies wurde später durch den Zuwachs an Wissen und Können sowie das Durchsprechen von Konzeptionen und damit frühzeitiger Erkennung von Schwachstellen oder Fehlern aufgewogen. Allerdings sollte unbedingt darauf geachtet werden, wer tippt – nämlich der Entwickler, der weniger weiß.

Auch bei Crystal Clear werden keine Techniken vorgeschrieben. Es gibt jedoch einige Vorschläge, die in der Projektdurchführung umgesetzt wurden. Hierzu zählt ebenfalls Pair Programming, aber auch die Blitzplanung am Anfang des Projekts. Diese empfand das Team insgesamt als sehr hilfreich für die Projektplanung, da sowohl Anwender als auch Programmierer anwesend waren und verschiedene Blickwinkel auf das Endprodukt einbezogen werden konnten.

Insgesamt gesehen wurden in allen Prozessen die Entscheidung getroffen, mit wenigen Prozessen zu beginnen und diese beherrschen zu lernen statt vieles gleichzeitig zu testen.

## Lieferung

Die Datum der Lieferungen unterschied sich bei allen drei Prozessen. Bei Scrum erfolgten sie wie in der Theorie angegeben am Ende jedes Sprints. Im Review konnte der Kunde das vorhandene Produkt auf seinem eigenen Rechner testen. Da dies ein Mac war und die Entwicklung auf Windows-PCs mit unterschiedlichen Auflösungen durchgeführt wurde, kam es insbesondere beim Layout öfter zu Unzufriedenheit und Ablehnung der User Story. Dies konnte im Rahmen des Projekts nur dadurch behoben werden, dass der Kunde seinen Computer für Tests zur Verfügung stellte. Der Sprint war jedoch so kurz, dass es meist schwierig war, den Kunden selbst für ein vorzeitiges Testen zu gewinnen. Ein weiterer negativer Aspekt der nur einwöchigen Iterationsdauer war die geringe Menge an User Stories. Falls der Sprint nur ein oder zwei User Stories umfasste, führt das Ablehnen durch den Kunden zu einer sehr niedrigen Velocity von bis zu 0. Dies ist demotivierend für das Team und macht die Planung des nächsten Sprints schwierig, bei der man sich an der Velocity orientiert.

Diese Nachteile gab es bei Crystal Clear nicht, da ein Release jeweils zwei Iterationen umfasste, also zwei Wochen. Außerdem ist der Anwender Teil des Teams und kann jederzeit zum Testen oder einer Besprechung herangezogen werden. Dies umzusetzen fiel dem Team jedoch unerwartet schwer, da das direkte Ansprechen des Kunden oder Anwenders eine völlig neue Erfahrung darstellte. Außerdem präsentierte das Team vorwiegend den Release, statt den Anwender noch einmal testen zu lassen, es hat also die Chancen der Kundenkommunikation und Rückmeldung noch nicht ausgeschöpft. Abgesehen davon ist die Blitzplanung eine sehr große Hilfe zur Organisation der Releases. Hier wird bereits festgelegt, welche User Stories vermutlich in welchen Release fallen werden. So hat der Kunde bereits zu Anfang eine gewisse Vorstellung über die Fertigstellung aller Features, was zu mehr Sicherheit seinerseits und in gewissem Maße mehr Zufriedenheit führte. Es sollte lediglich darauf geachtet werden, diese Planung nach jedem Release neu zu prüfen. Dem Kunden darf nicht das Gefühl gegeben werden, er könne an der Reihenfolge nichts mehr ändern. Dieser Fehler in der Kommunikation wurde bei der Durchführung des Projekts gemacht.

Kanban schließlich nutzte sowohl synchrone als auch asynchrone Iterations- und Release-Zyklen. Der asynchrone Zyklus fiel beiden Entwicklern sehr schwer, da ein Release mitten in der Iteration stattfand. Beiden war nicht klar, was sie während der restlichen Iteration tun sollten. Stattdessen wurden schließlich Releases nur ans Ende von Iterationen gelegt, aber die Anzahl der Iterationen war variabel. Dies stellte eine gute und sehr flexible Lösung dar.

## Änderungen der Anforderungen und der Arbeitsweise

Der Zeitpunkt, zu dem der Kunde ändernde Anforderungen kommunizieren durfte, war bei allen drei Prozessen entsprechend der Definition von Agilität in der Software-Entwicklung nicht eingeschränkt. Allerdings traten sie in allen drei Prozessen nur während der Vorstellung der Lieferungen auf, da hier der Kunde direkt mit dem Produkt in Kontakt kam. Ansonsten konnten insbesondere neue Anforderungen jeweils frühestens in der nächstfolgenden Iteration berücksichtigt werden, was den Kunden jedoch kommuniziert worden war. Abgesehen davon konnten bei Crystal durch das Einbeziehendes des Kunden und Anwenders ins Team direkt während der Iteration Änderungen vorgenommen werden. Insbesondere das Ablehnen von User Stories am Ende der Iteration wurde so teilweise verhindert.

Um die Arbeitsweise kontinuierlich zu verbessern, besitzen alle drei Prozesse Meetings zur ständigen Evaluierung. Bei Kanban insbesondere wurde der Operations Review am Ende eines Releases durchgeführt, was zum Teil zu sehr langen Zeiträumen ohne Veränderung führte. Obwohl es prinzipiell erlaubt und erwünscht ist, jederzeit Vorschläge zu machen, beispielweise zum Daily Standup Meeting, wurde dies so gut wie nie wahrgenommen. Auch hier liegt der Grund in der ungewohnt offenen Vorgehensweise. Die Entwickler waren es eher gewohnt, zu bestimmten Meetings Verbesserungsvorschläge zu machen. Außerdem betrafen einige dieser Ideen den Board-Aufbau, so z.B. das Einführen von Swimlanes, und diesen wollte man nicht mitten in der Iteration ändern.

Bei Crystal Clear und Scrum hingegen ist es tatsächlich nur vorgesehen, im entsprechenden Meeting Änderungen im Ablauf vorzuschlagen und zu beschließen. Der Reflexions-Workshop bei Crystal und die Retrospektive bei Scrum finden ebenfalls jeweils am Ende des Releases statt. Während die kurze einwöchige Phase bei Scrum als optimal empfunden wurde, waren die zwei Wochen bei Crystal bereits zu lang, um sich an Einzelheiten zu Beginn des Releases zu erinnern. Deshalb wäre es vielleicht besser, das Meeting nach jeder Iteration durchzuführen und damit einmal wöchentlich. Dieser kurze Rhythmus wurde von allen Teams begrüßt beziehungsweise für ihren Prozess vorgeschlagen. Auch eine Scrum-Projekt mit vierwöchigen Sprints sollte nicht nur alle vier Wochen eine Retrospektive abhalten. Dies ist zu unflexibel und erst recht zu lang, um sich an der Anfang der Iteration zu erinnern.

## Meetings und Kommunikation (schlimm, dass ich nicht alle Meetings namentlich erwähne?)

Sowohl die Umsetzung der verschiedenen Meetings als auch die Teamkommunikation gestalteten sich in den Prozessen sehr unterschiedlich. Prinzipiell wurden jedoch alle Evaluierungs-Meetings dazu genutzt, den jeweiligen Prozess für dieses Projekt und dieses Team zu verbessern und zu optimieren. Alle Teams hatten hierbei das Problem, dass sie nicht notwendigerweise im gleichen Raum waren, wenn die Dailies stattfanden. Dies war im Prinzip kein Problem solange alle das gleiche Board sehen.

Bei Kanban insbesondere wurden die Daily Meetings dazu genutzt, bereits Umsetzungsdetails für den aktuellen Tag zu klären. Das Triage-Meeting, welches regelmäßig prüft, ob User Stories entfernt werden können, weil sie zu alt oder schon erledigt sind, wurde hingegen nicht durchgeführt. Das Team sollte schließlich alle User Stories umsetzen. Allerdings wäre es hilfreich gewesen, um zu sehen, dass einige der vorhandenen User Stories bereits bei der Umsetzung anderer User Stories vollendet und damit überflüssig waren. Ohne dieses Meeting stellte das Team dies immer erst im Planning fest, wenn es darum ging, Tasks zu schreiben.

Bei Crystal Clear liefen alle Meetings wie gewünscht ab, allerdings entpuppte sich die Umsetzung der osmotischen Kommunikation als schwierig, da das Team meist räumlich getrennt war. Gleichzeitig benötigt man mindestens drei Personen, damit sie überhaupt funktioniert. Bei größeren Teams könnte sich diese Art der Kommunikation jedoch auch als störend für einige Teammitglieder erweisen.

Scrum schließlich hielt sich strikt an die Vorschläge für den Scrum-Prozess. Da beide Entwickler die drei Fragen im Daily meist übergingen, wurden sie als Pflichtaussagen mit Tabelle aufgenommen, die der Scrum Master täglich abhakte. Dies erleichterte die Kommunikation stark, da niemand zu sehr abschweifte und Unwichtiges erzählte. Außerdem wurde so das Daily kurz gehalten. Das Planning wurde mit wachsender Erfahrung kürzer und die Abschätzung mit Story Points genauer, so dass sich ein positives Gefühl im Team entwickelte. Dies gilt prinzipiell jedoch auch für die beiden anderen Prozesse.

## Dokumentation

Die Menge und Art der Dokumentation weicht in allen drei Prozessen voneinander ab. Den größten Aufwand hatte definitiv das Crystal-Team. Hier wurde am Anfang des Projekts die Grundsteinlegung erstellt, die bei der Organisation des weiteren Projekts jedoch sehr hilfreich war und den Chefdesigner als eine Art Ersatz-koordinator zur Seite stand. Ansonsten wurden lediglich ein Klassenmodell und das Design verschriftlicht. Diese sollten jedoch öfter auf Aktualität geprüft werden, sonst helfen sie wenig. Prinzipiell ist die Art der Dokumentation jedoch auch hier flexibel auswählbar; dies waren nur Vorschläge.

Dokumentation bei Scrum ist genauso freiwillig. Das Team dokumentierte lediglich die Retrospektiven und die daraus resultierende Definition of Done (DoD), die erklärt, wann ein Task geschlossen werden kann. Selbst das Burn-Down-Chart wurde bereits automatisiert vom Computer berechnet und nicht vom Scrum Master per Hand gezeichnet. Die DoD half besonders dabei, einen bestimmten Arbeitsablauf einzuhalten und führte zu mehr Abnehmen von Tickets durch den Kunden, und das Grundgerüst kann auch auf andere Projekte übertragen werden.

Das Kanban-Team schließlich dokumentierte gar nichts außerhalb des Kanban-Boards. Änderungen wurden sofort umgesetzt oder ans Board geschrieben. Der einzige Nachteil daraus entstand durch das Fehlen von Sprints in JIRA, so dass es schwierig war, auf vergangene Iterationen zurückzublicken und beispielsweise die früheren WIP-Limits oder Iterationslängen nur vage aus dem Gedächtnis bekannt waren. Dies würde das Team im nächsten Kanban-Projekt unbedingt schriftlich festhalten. Das Weglassen jeglicher Dokumentation wurde demnach als keine gute Arbeitsweise empfunden.

## Task-Größe und Aufwandsschätzung

Crystal und Scrum haben keine genauen Vorgaben zur Größe der Tasks oder User Stories. Beide Teams entschieden sich jedoch nach den ersten zwei bis drei Iterationen für möglichst kleine Tasks, da diese leichter fertig zu stellen und abzuschätzen waren. Beides erleichterte ebenso die Planung. Die Abschätzung bei Crystal Clear erfolgte bereits für User Stories und Tasks in der Blitzplanung zu Beginn und wurde später während den Planungsphasen für die Iterationen überarbeitet. Dies war nötig, da sich wie bei den übrigen Prozessen das Schätzungsvermögen des Teams ständig verbesserte. Trotzdem erleichterte die Vorabschätzung aus der Blitzplanung diesen Prozess. Falls ein Entwickler nicht sicher bei der neuen Abschätzung war, konnte er sich notfalls daran orientieren.

Bei Scrum wurden vor jedem Sprint im Sprint Planning User Stories abgeschätzt, indem die beiden Entwickler gleichzeitig eine bestimmte Anzahl Finger zeigten. Dies wurde alternativ zum Planning Poker durchgeführt, da keine Karten vorhanden waren. Für kleine Teams ist das eine gute Alternative, für größere wären vermutlich Karten besser, da man schnell den Überblick verliert und Entwickler so ihre Fingerzahl noch ändern können. Das ist aber nicht gewünscht. Auf Grund der Eigenheit von Yodiz musste das Team für dieses Board die Tasks auch abschätzen. Dies stellte sich als weitaus einfacher und akkurater heraus als die User Stories und wäre zumindest in neuen Teams, die das Abschätzen erst noch lernen, vielleicht eine gute Alternative zum Anfang. Dann müssen nur die Task Story Points addiert werden um zur User-Story zu gelangen.

Für Kanban sollten die Tasks möglichst gleich groß sein, so dass sie gleichmäßig durch das Board „fließen“ können. Dies stellte sich als nicht durchsetzbar für das Team heraus, insbesondere bei Pair Programming Tasks, die automatisch die doppelte Zeit beanspruchen. Deshalb wurden Swimlanes genutzt, um ähnlich große Tasks wie Tests zu gruppieren. Allerdings ging dadurch der Zusammenhang zu den User Stories verloren – also keine gute Idee in diesem Fall. Die Change-Request –Swimlane allerdings funktionierte mit einer Dauer von 30 Minuten pro Ticket sehr gut. Alternativ sollte mit SLAs statt Swimlanes gearbeitet werden, was jedoch leider in beiden Tools nicht vorgesehen ist und deshalb nicht vom Team getestet wurde. Die Aufwandsschätzung wurde bei Kanban direkt über Schätzung der Tasks und nicht der User Stories durchgeführt. Sie verbesserte sich wie bei den anderen Teams von Iteration zu Iteration und erlaubte es am Ende auch das nächste Release-Datum und damit effektiv drei Iterationen ziemlich akkurat zu schätzen.

## Priorisierung

Die Priorisierung ging bei allen Prozessen relativ gleich vonstatten. Der Kunde oder Anwender konnte prinzipiell am Anfang des Projekts eine Reihenfolge der User Stories festlegen. Diese konnte jederzeit geändert werden für alles, was nicht bereits in der laufenden Iteration bearbeitet wurde. Während bei Scrum die Abarbeitung des Boards von oben nach unten strikt eingehalten wurde, war dies bei Kanban nicht immer der Fall. Durch diesen Freiraum war es möglich, dass ein Teammitglied an einer User Story arbeitete, die es allein fertig stellen konnte, statt wie bei Scrum auf einen anderen Entwickler warten zu müssen, der behilflich ist. Dies war insbesondere bei flexiblen Arbeitszeiten sehr hilfreich, die sich nur zum Teil überschneiden, machten das Daily aber besonders wichtig für die Absprache.

## Empirie

Kanban ist eine Prozess, bei dem Empirie eine sehr große Rolle für den Verbesserungsprozess spielt. Allerdings stellte sich Yodiz als nicht hilfreich hierfür heraus, so dass lediglich JIRA verwendet werden konnte, um Bottlenecks im Fluss zu finden. Dies gelang beispielsweise als der Review der Tasks vom Team ignoriert wurde. In Burn-Up-Chart wäre insbesondere für die längeren Iterationen hilfreich um wünschenswert gewesen, um dem Team ein Gefühl für ihren Fortschritt zu geben.

Die beiden anderen Prozesse nutzten Empirie wenig bis gar nicht. Bei Scrum wurde das Burn-Down-Chart nicht zum Vergleich des Voranschreitens in der Iteration verwendet, obwohl es vorhanden war, da JIRA lediglich abgeschlossene User Stories anzeigt und deren Anzahl für den sehr kurzen Sprint und das sehr kleine Team entsprechend gering war. Bei größeren Teams und eventuell längeren Sprints hingegen ist das Diagramm ein sehr hilfreiches Tool. Zumindest die berechnete Velocity nutzte das Team aber für die Planung des nächsten Sprints. Crystal nutzte keinerlei empirische Methoden während des Projekts. Trotzdem funktionierten das Team und die Planung sehr gut. Möglicherweise halfen hier andere Maßnahmen wie beispielsweise die Blitzplanung oder auch die gute Absprache zwischen den Teammitgliedern.

## Kundenkontakt

Der Kundenkontakt stellte sich bereits als erstaunlich schwierig heraus, wenn man bedenkt, dass in diesem Fall für alle drei Projekte die Kunden genauso oft verfügbar waren wie die Entwickler. Trotzdem gab es in allen Gruppen lediglich Kommunikation mit dem Kunden zum Release-Datum. Dies war besonders bei Kanban und Crystal während der längern Zyklen zu selten. Allerdings sollten insbesondere bei Crystal der Kunde und der Anwender ständig gefragt werden können – sie sind schließlich Teil des Teams. Aber auch bei den beiden anderen Prozessen ist die Kommunikation mit dem Kunden während der Iteration erlaubt und erwünscht. Darauf sollte unbedingt bei der Umsetzung aller drei Prozesse in neuen Teams geachtet werden, da es ein Umdenken zum „alten“ Vorgehen erfordert, bei dem ich als Entwickler vielleicht zusammen mit anderen Entwicklern entscheide, was der Kunde wahrscheinlich gemeint haben könnte.

## Fazit (NAME FÜR DIE ÜBERSCHRIFT?)

Insgesamt gesehen hatte jeder Prozess Vor- und Nachteile, die oft mit der Teamgröße und Vorerfahrung zusammenhingen. Aus den gemachten Erfahrungen und dem theoretischen Vorwissen über die Vorgehensweisen, entstand die folgende Einschätzung dafür, wann welcher Prozess gut geeignet ist.

Crystal Clear befindet sich durch die Blitzplanung und die Dokumentation mittels Grundlegung noch recht nahe am „nicht-agilen“ Projektmanagement und bietet somit vielleicht einen guten Einstieg in Agilität – sowohl für Kunden als auch für Entwickler. Die Dokumentation ist nicht plötzlich weggefallen und auch eine Art Planung des gesamten Projekts ist vorhanden und gibt eine gewisse Sicherheit für alle Beteiligten. Das Team sollte jedoch nicht zu klein gewählt werden und man benötigt für osmotische Kommunikation tatsächlich Mitarbeiter von Kundenseite, die mit den Entwicklern im selben Raum sitzen – und zwar langfristig. Dies ist wohl am ehesten möglich bei innerbetrieblichen Projekten.

Scrum hingegen eignet sich eher für einen Einstieg, bei dem Kunde und Entwickler damit zufrieden sind, dass nicht alles vollständig vorher geplant und dokumentiert ist. Auch hier ist eine gewisse Mindestgröße des Teams nötig, um die geforderten Rollen auszufüllen. Dafür muss der Kunde und Anwender nicht ständig mit den Entwicklern im selben Raum oder Gebäude sein, solange anderweitige Kommunikation z.B. via Telefon möglich ist. Der geregelte Ablauf in Scrum, d.h. die gleich bleibende Iterationslänge, die verschiedenen Meetings und Artefakte, ist gut geeignet für Einsteiger in agiles Projektmanagement. Sie helfen dabei, sich in die selbstorganisierte Arbeitsweise einzufinden und mit der recht großen Planungsfreiheit umzugehen.

Kanban kann sich ebenfalls als Einstieg in agiles Prozessmanagement eignen, allerdings nicht sofort mit all seinen Freiheiten und Möglichkeiten. Dinge wie asynchrone Iterationen und Releases erfordern sehr viel Erfahrung mit dem Ablauf und sind für Einsteiger eher ungeeignet. Das WIP-Limit und ein einfaches Kanban-Board hingegen, welches praktisch dem Scrum-Board gleicht, sind sehr gute Einstiegspunkte. Da Kanban prinzipiell dazu dient, den bestehenden Prozess stufenweise zu verändern, kann es jedoch auch in einem Team genutzt werden, das bereits andere agile Techniken wie Scrum oder Crystal einsetzt. Dann können auf Grund der Vorerfahrung auch gleich fortgeschrittene Techniken wie Swimlanes eingesetzt werden. Was insbesondere für den Einsatz von Kanban spricht, ist also eine Teamstruktur, bei der der Prozess eher langsam geändert werden soll, statt ihn komplett zu ersetzen. Außerdem funktioniert Kanban sowohl bei kleinen als auch großen Teams sehr gut, insbesondere wenn verschiedene Mitglieder spezifische Aufgaben erfüllen, wie beispielsweise Konzeptersteller, Programmierer oder Tester.