Basisausbildung Post – Project Training Journal

Im Rahmen eines Kleinprojekts mit mehreren Entwicklern eine vereinfachte Form von SCRUM umsetzen.

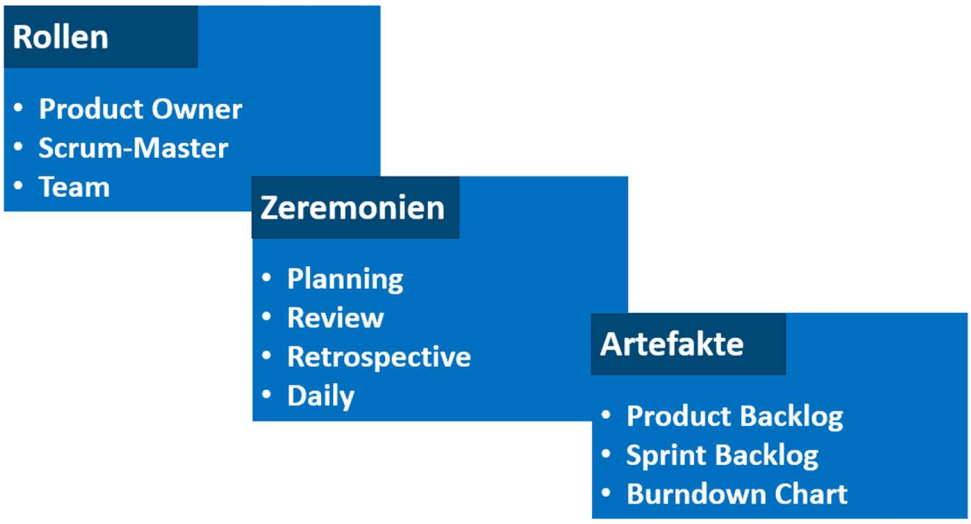
Dieses Journal wird während dem gesamten Modul gepflegt. Es werden täglich Einträge verfasst. Ziel dieser Dokumentation ist das aktive Reflektieren der Projektmethodik und dessen Erfolgen / Misserfolgen. Die folgenden Modulziele gelten für die gesamte Modulzeit.

MZ1:	Alle Teilnehmenden setzen im Rahmen eines Kleinprojekts mit mehreren Entwicklern eine vereinfachte Form von SCRUM um
MZ2:	Alle Teilnehmenden kennen die Begriffe SCRUM, Sprint, Artefakt, BackLog, Daily, Retrospektive und können diese selbstständig erklären
MZ3:	Alle Teilnehmenden kennen die vorhandenen SCRUM Rollen und erklären deren Funktion und Aufgaben selbstständig
MZ4:	Alle Teilnehmenden nehmen während dem Modul mindestens einmal die Rolle des Entwicklers und des SCRUM Masters ein und treffen auf Grundlage des Aufgabenportfolios selbstständig Entscheidungen zugunsten des Projektes.
MZ5:	Alle Teilnehmenden können mittels «Story Points Schätzen» realistische Aufwandsschätzung für simple User Stories im Rahmen des Settings treffen
MZ6:	Alle Teilnehmenden protokollieren den Erfolg und Misserfolg der eingesetzten Projektmethodik in einem Journal

Inhaltsverzeichnis

SCRUM in eigenen Worten

SCRUM ist eine gängige und sehr bekannte Projektmethodik. SCRUM wird in unterschiedlichen Firmen, Bereichen und Abteilungen unterschiedlich eingesetzt. Im ICT-Campus lernen wir eine reduzierte, vereinfachte Form von SCRUM kennen, setzen aber viele Grundfunktionalitäten um.



Projektrollen

Product Owner

… ist entweder der Kunde selbst oder eine Person aus der eigenen Firma, welche den Kunden repräsentiert. In letzterem Fall spricht diese Person sich regelmässig mit dem Kunden ab, um sicherzustellen, dass sie die Vorstellungen sehr genau kennt.

Der Product Owner nimmt die fertiggestellten Arbeitspakete ab. Wenn etwas noch nicht seinen Vorstellungen entspricht, lehnt er das Arbeitpaket ab und es muss im nächsten Sprint erneut bearbeitet werden.

Es empfiehlt sich, im Planning und während dem Implementieren regelmässig den Product Owner mit Fragen zu löchern. Damit stellt man sicher, dass die Umsetzung genau nach seinen Vorstellungen und Anforderungen abläuft und das Arbeitspaket im Review auch abgenommen wird.

Scrum-Master

Diese Person kümmert sich um das Befolgen der Scrum Methodik. Es kann ein Entwickler aus dem Team diese Rolle zusätzlich innehaben oder in grösseren Teams ist eine Person nur Scrum Master.

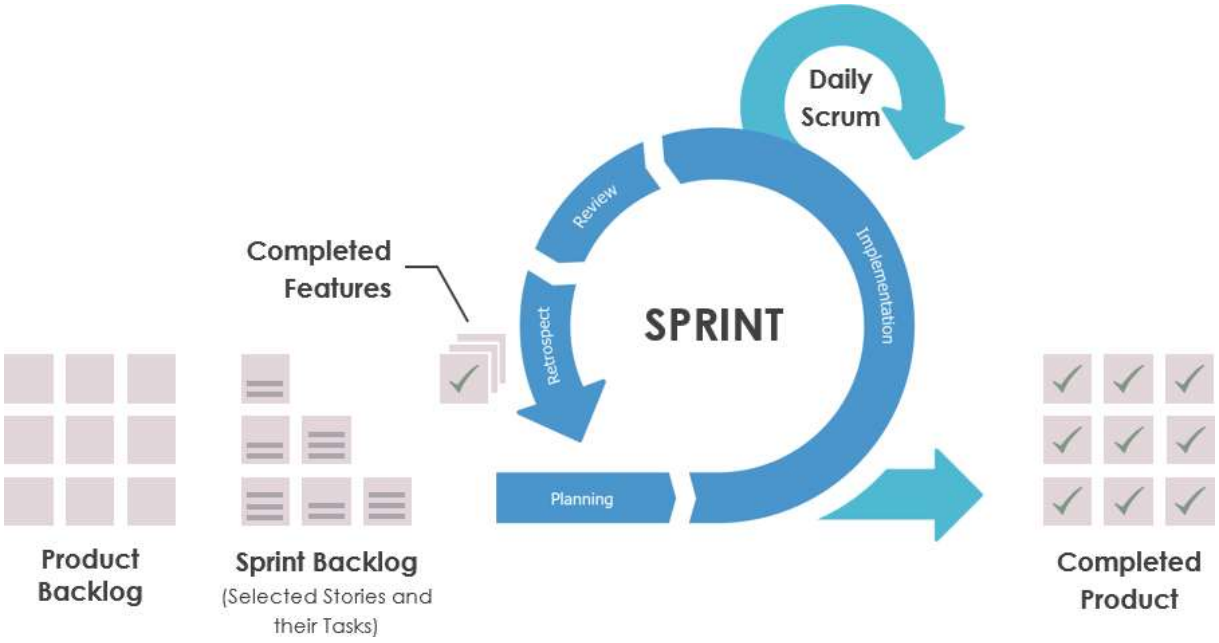
Der Scrum Master ist dafür verantwortlich, dass die Zeitvorgaben von Meetings eingehalten werden, dass das Team sich reflektiert und selbst verbessert und schützt das Team von anderen Arbeiten (Impediments / Hindernissen).

Entwickler (Team)

Entwickler-Team besteht aus 5-9 Personen (maximal 10 Personen inkl. PO), wobei oft viele verschiedene Funktionen vertreten sind (Design, Entwicklung, Testen etc.) – es sind meistens interdisziplinäre Teams. Teams organisieren sich selbst. Ein Teammitglied ist normalerweise (idealerweise) nur in einem Team tätig, damit keine Ablenkungsgefahr besteht.

Sprint

Als Sprint wird eine Scrum-Druchführung (ein Durchgang) bezeichnet. Wenn ein Projekt nach Scrum umgesetzt wird, wird es in Sprints unterteilt. Dabei dauert ein Sprint normalerweise 2 Wochen. Der Spring beginnt mit einem Planning, hat jeden Morgen ein Daily Stand Up und schliesst mit Review und Retro ab. Im Normalfall schliesst das Planning des nächsten Sprints direkt an das Retro an.



Ereignisse / Zeremonien

Spring Planning

Teilnehmende: Team, Scrum-Master und Product Owner

Das Team entscheidet gemeinsam, wie viele Tasks es im anstehenden Sprint schaffen kann. Der Product Owner bringt die Prioriäten mit («was?») und das Team legt fest, was es schaffen kann («wie viel?»).

Gemeinsam spielt das Team dann Planning Poker. Ein Spiel, bei die Teammitglieder jeden Task aus dem Sprint Backlog (Sprint Backlog wird vom PO gefüllt) anschauen und mit einem Komplexitätsscore den Arbeitsaufwand abschätzen. Der Scrum Master präsentiert einen Task und die Teammitglieder schätzen verdeckt, wie viele Story Points (Komplexität, Aufwand, Ungewissheit/Risiko) dieser Task hat. Für die Zuteilung wird die Fibonacci-Folge verwendet, da je grösser die Komplexität, desto grösser auch die Unsicherheit bei der Einschätzung.

Dann werden alle Voten gleichzeitig aufgedeckt. Bei grosser Diskrepanz kann kurz diskutiert werden. Danach einigt man sich auf eine Zahl. Diese wird dem Task dann zugewiesen.

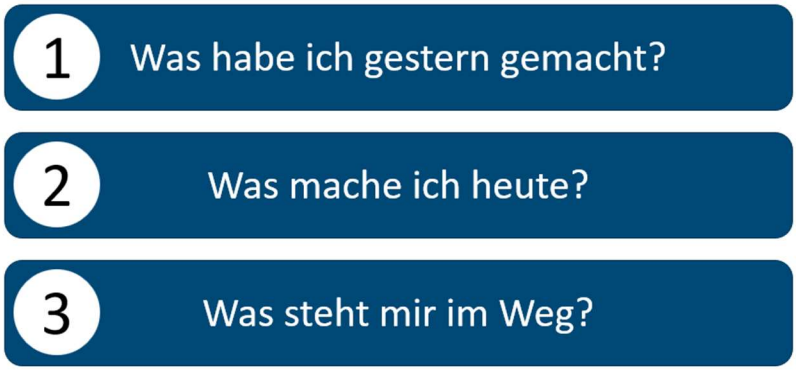
Die Tasks bleiben aber ohne Zuweisung an ein Teammitglied. Dies geschieht erst während dem Sprint.

Implementierung

Während der Implementierung nimmt sich jedes Teammitglied einen Task und führt diesen aus. Wenn der Task beendet ist, nimmt sich das Teammitglied einen weiteren Task. Man kann nur einen Task aufs Mal nehmen. Es können keine Tasks reserviert werden.

Daily Standup

Einmal pro Tag, typischerweise am Vormitag, setzt sich das Team zusammen und jeder beantwortet kurz folgende Fragen.



Dabei werden keine Challenges beim Entwickeln detailliert besprochen, das kann man bilateral lösen. Allgemein wird nicht diskutiert, sondern nur informiert. Das Ziel ist es, dass alle Teammitglieder voneinander wissen, wer and was arbeitet. Das verhindert auch, dass zwei Personen aus Versehen dasselbe Feature umsetzen.

Review (Produkt)

Hier wird die Arbeit des Sprints dem Product Owner (ggf. auch in Anwesenheit des Kunden) präsentiert. Dies ist keine PowerPoint-Präsentation, sondern ein tatsächliches Ausführen des Codes.

Idealerweise hat der PO die Tasks schon während des Sprints geprüft, damit hier keine Überraschungen entstehen.

Retrospect / Retro (Prozess)

Hier wird methodisch reflektiert: Wie haben wir gearbeitet? Was lief gut? Was müssen wir ändern? Was können wir so weiterführen?

Das Ziel des Retros ist es, dass das Team sich verbessert und im nächsten Sprint noch erfolgreicher zusammenarbeiten kann.

Artefakte

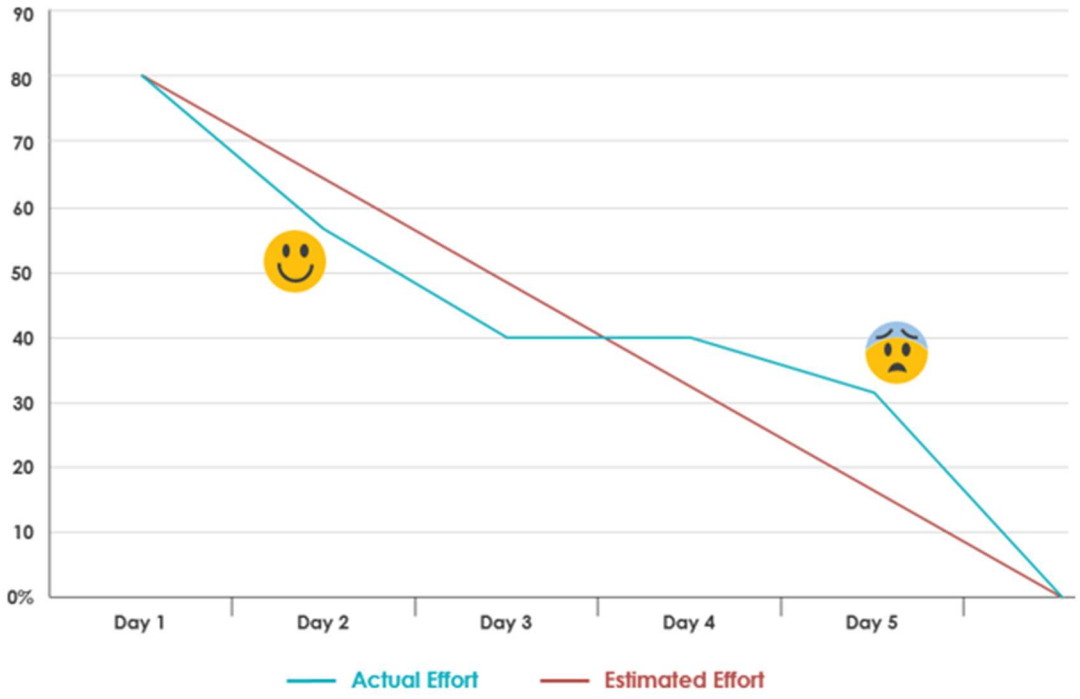
Product Backlog

Der Product Owner Teilt das Projekt in grobe User Stories / Backlog Items auf. User Stories beschrieben das zu erreichende Outcome und helfen beim Verständnis, was ein Feature genau tun soll. In Vorbereitung auf das Sprint Planning priorisiert der Product Owner die Tasks aus dem Product Backlog, er entscheidet, welche Tasks für diesen Sprint anstehen.

Sprint Backlog

…umfasst alle Tasks für den aktuellen Sprint und kann nach Abschluss des Plannings nicht mehr verändert werden (ausser wenn das Team merkt, dass ein Task wichtig zur Erreichung des Sprint-Ziels ist, kann es diesen nachträglich noch mit aufnehmen). Die Tasks haben auch innerhalb des Sprint Backlogs eine Priorisierung. Diese gibt einen grobe Richtung für die Teammitglieder, welche Tasks sie während der Implementierung zuerst bearbeiten sollten.

Burndown Chart



Das Burndown Chart zeigt die noch verbleibende Arbeit innerhalb eines Sprints anhand einer gerade Linie vom zeitlichen Startpunkt des Sprints und der Höhe der gesamten Story Points des Sprints hin zum Endpunkt des Sprints und dem Ziel (0 Story Points).

Die tatsächliche Kurve zeigt dabei jeweils, ob das Team «on track» ist oder nicht. Es ist üblich, dass man am Anfang hinterherhinkt und gegen Ende aufholt. Dies kommt daher, weil es immer eine gewisse Anlaufszeit braucht, um in den Flow zu kommen.

Vorbemerkung zur Nummerierung der Tage

Es gab eine Verwirrung, wodurch ich nun bei Tag 1 nicht einen kompletten Journaleintrag habe. Ich habe Tag 1 (Start des ersten Sprints) und Tag 2 (der erste Sprint) nun als einen gemeinsamen Journaleintrag hier festgehalten. Ich führe einen Eintrag pro Sprint. Dabei verstehe ich ein Tag jeweils Synonym mit einem Sprint. Auch wenn der Sprint offiziell bereits kurz vor Ende des Vortags schon beginnt…

Sprint 1 – Mittwochabend 11. März + Donnerstag 12. März

Wie ist das Planning aus deiner Sicht gelaufen?

Das Planning ist gut gelaufen: Wir haben ca. 1 Stunde gehabt, was sehr lange ist. Allerdings war es unser erstes Meeting mit dem Kunden (PO Michael Moser, unser Coach). Entsprechend kannten wir die Vision noch nicht und kannten kaum Details zum Projekt.

Nach diesem Planning Meeting haben wir nicht nur die Tasks für den ersten Sprint definiert, sondern auch sehr detailliert mit dem Kunden die Web-App ausgearbeitet in Bezug auf seine Anforderungen, wie er sich diese vorstellt.

Besonders spannend fand ich das Tool Scrum Poker Online (https://planningpokeronline.com/), welches einem Team erlaubt, Scrum Poker zu spielen und die Schätzung für die Story Points eines jeden Tasks auch tatsächlich verborgen abzugeben. Das war sehr spannend.

Wir hatten, während dem Scrum Poker, einige Punkte, in denen wir uns nicht einig waren, bzw. die Diskrepanz zwischen den abgestimmten Story Points sehr gross war. Ich war dabei immer etwas auf der höheren Seite. Dies einfach, weil aus meiner Erfahrung die ersten Tasks im Projekt (Konzeption und Planung, Corporate Design, ERD, Backend-Struktur etc.) viel Gewicht bekommen sollten. Meiner Meinung ist jede Minute, die wir hier Investieren, eine Stunde, die wir bei der Umsetzung sparen.

Bist du zu zufrieden mit den dir zugeteilten Tasks?

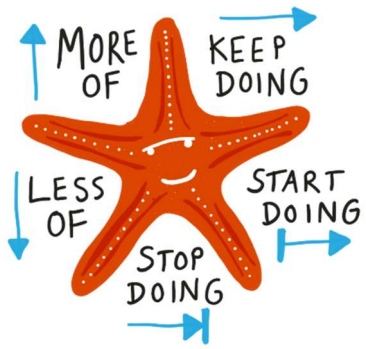
Ich habe mich um die Ausarbeitung des Corporate Designs gekümmert. Damit bin ich zufrieden, da ich denke, dass ich damit einen wichtigen Grundstein legen kann. Wenn das Corporate Design sauber ausgearbeitet ist, dann fallen die weiteren Designschritte deutlich leichter und harmonischer. Da diese sich auf eine Single-Source-Of-Truth beziehen können, statt dass jeder seine eigene Version von unserer Webapp ausdenkt und wir dann ein grosses Chaos haben.

Stimmt deine User Storie Schätzung?

Die Schätzungen waren zum Teil nicht korrekt. Dies lag oft daran, dass wir im Planning die Tasks nicht genau ausformuliert hatten. Entsprechend wussten wir zum Teil nicht genau, was alles dazugehörte und was nicht. Ich denke, es ist wichtig, im Planning die Tasks sehr detailliert zu beschreiben.

Hat die Retrospektive einen Gewinn für die Gruppe ermöglicht? Welchen? Wie hast du die Wahl der Retro-Methode empfunden?

Die Starfish-Methode war gut gewählt. Wir konnten einige Punkte finden, die wir vermehrt tun wollen und konnten auch viele positive Dinge nennen, welche schon sehr gut geklappt hatten am ersten Tag.



Der wichtigste Gewinn war, dass wir erkannt haben, dass wir bei Fragen zuerst selbst recherchieren sollen, bevor wir die anderen in deren Arbeit unterbrechen.

Wie hat der Scrum Master heute gearbeitet?

Er hat das Retro gut vorbereitet, in dem er auf der Pin-Wand einen Starfish gezeichnet hat. Beim nächsten Mal könnte Kevin einen dickeren Stift und mehr Zettel verwenden, um die Dinge an der Pinwand dann zu notieren, da einfach mit Kugelschreiber es zu klein ist, um zu lesen von weit weg. Zudem könnte man diese Zettel ggf. dann gerade hängen lassen für den nächsten Sprint, damit man an die Action Items erinnert wird.

Betrachte die Modulziele auf der ersten Seite dieses Journals. Welche Ziele konntest du bisher erreichen?

MZ2:	Alle Teilnehmenden kennen die Begriffe SCRUM, Sprint, Artefakt, BackLog, Daily, Retrospektive und können diese selbstständig erklären

Heute habe ich dieses Ziel erreicht. Ich kann jetzt die grundlegenden Scrum-Begriffe erklären.

Key Learnings

Das Vorgehen des Kunden (PO) darf man auch kritisch hinterfragen.

Wir haben im heutigen Planning viel zu sehr die Tasks des PO einfach für bare Münze genommen. Heute habe ich gemerkt, dass unser PO (fikitv! Aber er spielt es sehr gut…) kein Entwickler ist und daher eigentlich gar nicht weiss, wie das Projekt abzulaufen hat! Er präsentiert zwar stolz seine Tasks (Datenbank ersetellen etc.) und will dann direkt, dass man es mit Story Points versieht etc., doch das muss man durchaus auch kritisch bewerten: Gibt es da noch andere Tasks aus Entwicklersicht, die wir mit in den Sprintbacklog aufnehmen sollten? So hatte er zum Beispiel keinen Task zu Anforderungen definieren, Use-Case-Diagramm erstellen und ERD erstellen geplant. Absolut grundlegende Schritte, die wir unbedingt umsetzen sollten, ehe man direkt in den Code springt (das wäre nämlich ein Anfängerfehler!). Ich will morgen, wo unser erster Sprint so richtig losgeht, das noch einbringen, dass wir diese Dinge unbedingt tun sollten!

Wobei nun, da ich das so schreibe fällt mir auch auf, dass das eigentlich wieder genau nicht-scrum, sondern Wasserfall ist. Das, was wir zwar vorher gelernt haben, aber jetzt wohl wieder etwas vergessen sollten? Ich finde das aktuell noch sehr schwierig zu verstehen, wie viel Wasserfall und wie viel agil. Beziehungsweise: Wie viel Wasserfall braucht das Agile? Ich denke, dass ist meine Kernfrage von heute.

Sprint 2 – Freitag, 13. März

Wie ist das Planning aus deiner Sicht gelaufen?

Das Planning verlief deutlich konstruktiver als am Vortag. Eine wesentliche Verbesserung war die Detaillierung der Tasks direkt in Trello. Indem ich technische Details und Abgrenzungen sofort dokumentiert habe, ist die Erwartungshaltung und Aufgabenbeschreibung für den Sprint klarer definiert. Das mindert die Gefahr, über die Definition of Done hinauszuarbeiten versehentlich.

Besonders im Hinblick auf die Zusammenarbeit mit weniger erfahrenen Teammitgliedern erwarte ich hier eine Zeitersparnis: Im letzten Sprint führten vage Task-Beschreibungen zu häufigen Unterbrechungen meines eigenen Workflows, da grundlegende Anforderungen erst im Nachgang erklärt werden mussten.

Prozess-Optimierung für die Zukunft: Wir haben vereinbart, bei Schätzabweichungen von nur einer Stufe (einer Karte) auf eine erneute Diskussion und Neuschätzung zu verzichten, um den Prozess schlank zu halten. Nur bei kontroversen Einschätzungen gehen wir tief in die Analyse.

Bist du zufrieden mit den dir zugeteilten Tasks?

Ja, die Aufgabenmischung ist sehr motivierend. Das Troubleshooting im Backend ist zwar anspruchsvoll, bietet aber einen hohen Lerneffekt. Zudem war die Umsetzung der zentralen api.js im Frontend ein Erfolgserlebnis, da sie eine wichtige strukturelle Basis für das Projekt bildet.

Stimmt deine User Story Schätzung?

Die Schätzungen sind grundsätzlich präziser geworden, allerdings zeigt sich eine Tendenz zur "Unter-Schätzung", da vieles «dann doch länger geht als gedacht». Wir müssen zudem lernen, die Zeit für Diskussionen zu begrenzen, wenn die Tendenz der Gruppe bei User Story Schätzungen ohnehin in eine ähnliche Richtung weist.

Ein kritisches Thema bleibt die Scope-Definition: Ein Task, der "das gesamte Backend" umfassen soll, ist für einen Sprint zu groß und gefährdet die Lieferfähigkeit. Hier müssen wir in Zukunft feingranularer schneiden.

Hat die Retrospektive einen Gewinn für die Gruppe ermöglicht? Welchen? Wie hast du die Wahl der Retro-Methode empfunden?

Die Retrospektive war wertvoll, um interne Kommunikationslücken aufzudecken. Besonders der Umgang mit Ad-hoc-Aufgaben wurde konstruktiv kritisiert. Es wurde deutlich, dass Aufgaben nicht "auf Zuruf" oder eigenmächtig ohne entsprechendes Ticket im Tool gestartet werden sollten. Die gewählte Methode half dabei, die Frustration über ungeplante Tasks sachlich zu adressieren und Verantwortlichkeiten (z. B. wer ein Ticket erstellt, wenn man in ein Meeting muss) klarer zu benennen.

Zudem haben wir heute gemerkt, dass wir viele "Loose Ends" haben: Viele angefangene, aber nicht abgeschlossene Features. Ich habe dazu recherchiert und gemerkt: In der agilen Welt gilt: "Stop starting, start finishing." Ein zu 90% fertiges Feature hat einen Lieferwert von 0! Das ist sehr wichtig zu beachten. Bis heute (bereits die Hälfte des Projekts) haben wir noch nichts «geliefert»!

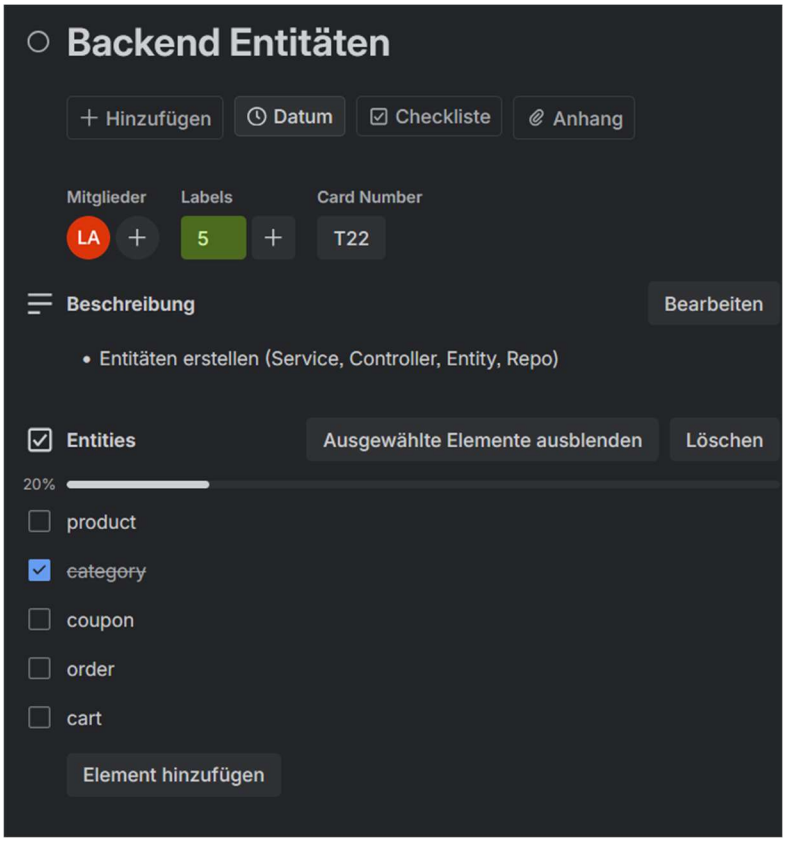
Geschwindigkeit ist nicht dasselbe wie Lieferfähigkeit. Wir haben zwar schon 5 Unterseiten begonnen, aber noch kein einziges nutzbares Feature für den User abgeschlossen.

Wie hat der Scrum Master heute gearbeitet?

Es stellt sich die Frage der Kontrollinstanz: Es gab Situationen, in denen Teammitglieder ohne aktiven Task an Code-Teilen gearbeitet haben, die bereits im Review-Status waren. Hier wäre ein proaktiveres Eingreifen des Scrum Masters wünschenswert gewesen, um sicherzustellen, dass jede Arbeitsstunde auch einem sichtbaren (und geschätzten) Task zugeordnet ist. Hier habe ich dann eingegriffen, als es mir aufgefallen ist, auch wenn es nicht meine Aufgabe war. Die Überwachung des Board-Status und die Moderation bei personellen Engpässen (z. B. wenn jemand durch fehlende Frontend-Strukturen blockiert ist) könnten noch konsequenter erfolgen.

Mal eben das ganze Backend gebaut?

Ich finde es sehr bemerkenswert, dass eine Person mit diesem Task hier…



…mal fast das komplette Backend gebaut hat. Die Person hat an diesem 5er-Task 1.5 Arbeitstage lang gearbeitet. Weil dieser Code dann immer nur bei dieser Person lokal war und nicht auf dem main, hat uns das einen ganzen Tag lang länger blockiert mit Frontend-Anbindungen als nötig. Das hat uns massiv in unsere Lieferfähigkeit in diesem Sprint eingeschränkt.

Wie löst man das? Ich hatte dieses Problem auf dem Radar und habe die Person vermehrt gefragt «mal zum Punkt zu kommen», wonach es aber nochmal 2-3 Stunden dauerte.

Hier wäre es wohl die Aufgabe des Scrum Masters gewesen, zu schauen, dass dieses Nadelöhr möglichst bald gelöst wird. Der Scrum Master hatte heute aber nicht wirklich auf dem Schirm wer gerade was macht, sondern ist eher einfach nur seinen formellen Aufgaben (z.B. Retro planen und durchführen) nachgegangen.

Repo-Problem /Frontend vs. /frontend

Auf dem GitHub-Repository (Linux-Basis) existierten zwei Ordner, die sich nur durch die Gross-/Kleinschreibung unterschieden: Frontend/ und frontend/. Beim Klonen auf unsere Windows-Rechner trat ein Fehler auf: Windows ist standardmässig case-insensitive. Das heisst, Windows „sieht" keinen Unterschied zwischen den beiden Namen und legt lokal nur einen der beiden Ordner an. Folge: Dateien im „unsichtbaren" Ordner fehlten lokal, Importe im Code waren defekt und Git zeigte einen inkonsistenten Status an.

Learnings

Naming Convention: In Projekten sollten Ordner- und Dateinamen **immer kleingeschrieben werden** (lowercase-only). Das vermeidet OS-übergreifende Konflikte von vornherein.

Windows-Features: Windows kann (seit Win 10) pro Verzeichnis Case-Sensitivity, aber es ist ein „Opt-In"-Feature für Spezialfälle wie diesen und ist nur fürs Debugging von solchen Problemen geeignet, da Ordner so schnell mal nicht mehr normal funktionieren.

Git-Config: Die Einstellung core.ignorecase ist unter Windows standardmässig auf true, was in 99% der Fälle gewollt ist, aber bei Casing-Konflikten manuell korrigiert werden muss.

Wenn Dateien lokal fehlen, die auf GitHub sichtbar sind: Sofort das Casing prüfen! Bevor ich lokal bastle, ist es oft schneller, den Ordner direkt auf der GitHub-Weboberfläche umzubenennen oder zu verschieben, um den Konflikt dort zu lösen, bevor er auf die Windows-Maschine kommt.

Schlusswort

Dieses Problem hat Luna und mich für ca. 2 Stunden beschäftigt. Also ca. 4 Arbeitsstunden sind hier draufgegangen. Mich erstaunt, dass sowohl Senior Developer Luna und Senior Senior Developer Leandro diverse PRs durchgewunken haben, die diese Dualität /frontend /Frontend hatten. Haben die beiden nicht genau wegen solchen Gefahren ihre Rolle inne?

Betrachte die Modulziele auf der ersten Seite dieses Journals. Welche Ziele konntest du bisher erreichen?

MZ1: done	Alle Teilnehmenden setzen im Rahmen eines Kleinprojekts mit mehreren Entwicklern eine vereinfachte Form von SCRUM um
MZ2: done	Alle Teilnehmenden kennen die Begriffe SCRUM, Sprint, Artefakt, BackLog, Daily, Retrospektive und können diese selbstständig erklären
MZ3: done	Alle Teilnehmenden kennen die vorhandenen SCRUM Rollen und erklären deren Funktion und Aufgaben selbstständig

Ich konnte die ersten drei Ziele bereits erreichen. Im ersten und zweiten Sprint ist mir sehr klar geworden, was die Rollen bedeuten und welche Funktionen und Aufgaben sie haben.

Key Learnings

Wie viel Wasserfall braucht Agile?

Diese Frage ist mir beim Lernjournalschreiben erst aufgefallen. Das ist der Hauptkonflikt aktuell in meinem Kopf. Mein Kopf will die Web App perfekt nach Wasserfall aufbauen: Zuerst alle Anforderungen bis ins letzte Detail aufschreiben, dann Use-Case-Diagramme zeichnen, dann 50 API-Endpoints definieren etc. – Doch machen wir dann noch Scrum? Dann machen wir Wasserfall, aber als Scrum getarnt!

Diese Grundsatzfrage hat mich heute umgetrieben. Und ich habe mit KI ein Gespräch darüber begonnen. In diesem Gespräch ist mir bewusst geworden, dass im Scrum offenbar das Prinzip «Just enough planning» gilt. Dabei muss man Scrum als Interaktion zwischen Wasserfall und Agile verstehen. Mit Scrum arbeiten wir grundsätzlich agil. Wir wollen bereits von Anfang an kleine Features zeigen können. Selbst der allererste Sprint hat bereits viel Coding drin und nicht sehr viel Konzeptionelles. Man macht nur so viel konzeptionelle Arbeit, wie es eben gerade braucht. So macht man das ERD gerade so gross, wie man es für das erste Feature braucht (z.B. Login). Dafür brauchen wir nur eine Tabelle: Die Tabelle User. Und dabei bleibts auch! Sehr spannend. Das hätte ich nicht gedacht, dass man ein Projekt so konsequent agil machen kann. Ich stelle es mir aber auch sehr herausfordernd vor: Wenn man nur Feature um Feature plant und umsetzt, dann läuft man doch zwangsläufig irgendwann ins Messer, oder? Man baut doch dann das Puzzle, ohne das Bild zuerst gesehen zu haben? Können da nicht grosse Architektur-Konflikte entstehen mit der Zeit?

Ich hatte heute, nachdem wir im ersten Sprint keine grossen Konzepte gemacht haben, grosse Bedenken, dass wir ohne Gesamt-Architektur in eine Sackgasse steuern. In einem ausführlichen Gespräch mit der KI ist mir bewusst geworden, dass Scrum darauf vertraut, dass die Architektur durch ständiges Refactoring und eine klare Vision des PO mitwächst. Die Architektur selbst ist also auch agile! Was für eine Erkenntnis… Bisher dachte ich immer: Ok, wir wissen jetzt wie man ein Softwareprodukt sauber durchplant, jetzt setzen wir halt die Features nach Scrum um, statt nach Wasserfall. Aber nein, mein Freund! Die Konzeption und Planung selbst sind auch agil! Sehr spannend.

Die Herausforderung ist also nicht, von Anfang an alles perfekt zu planen, sondern **den Code so flexibel zu halten**, dass er sich verändern lässt. Selbst das ERD kann am Anfang aus nur einer Tabelle bestehen (z.B. User) für das Login. Sobald das da ist und man z.B. Produkte erfassen will, kommt die Tabelle «products» dazu. Das scheint mir jetzt noch sehr «abenteuerlich», doch das ist wohl genau die Idee dieses Moduls: Einfach mal machen und wir werden dann schon sehen, wo (*wenn überhaupt?*) wir an «fehlendem Architekturdesign» scheitern.

Um die Einstiegsfrage zu beantworten: Agile braucht nur immer genau so viel Wasserfall (Planung), bis man das aktuell zu entwickelnde Feature coden kann.

Just enough planning

Spring 3 – Mittwoch,18. März

Wie ist das Planning (Freitag) aus deiner Sicht gelaufen?

Das Planning war geprägt von grossen Fortschritten. Eine wichtige Erkenntnis war jedoch der Umgang mit der Definition of Done (DoD): Wir haben festgestellt, dass Teammitglieder (mich eingeschlossen) oft über die eigentlich gedachte DoD hinausarbeiten. Da Änderungen durch zwei Instanzen müssen, bevor sie auf dem main landen, ist Geschwindigkeit hier entscheidend.

Fazit für die Zukunft: Tasks müssen im Planning noch detaillierter beschrieben werden. Die Definition of Done sollte idealerweise direkt im Task dokumentiert sein, um Fehlinterpretationen zu vermeiden und den Merge-Prozess zu beschleunigen.

Bist du zu zufrieden mit den dir zugeteilten Tasks?

Ja, wobei sich mein Fokus stark verschoben hat. Ich übernehme aktuell viel High-Level-Arbeit, Coaching und strategische Themen (z. B. Architektur-Entscheidungen zum Bilderupload, Vereinheitlichung der Frontend-Pfade etc.).

Im Austausch mit Coach Michi konnte ich klären, wie diese "Senior-Aufgaben" in Scrum abgebildet werden. Obwohl Scrum von gleichberechtigten Entwicklern ausgeht, übernehmen in der Praxis oft erfahrene Teammitglieder diese steuernden Aufgaben parallel zur Entwicklung. Zudem haben wir gelernt, dass High-Level-Aufgaben im Code (wie das Refactoring aller Pfade) als explizite Tasks im Planning erfasst werden müssen, um Blindheit für das "Grosse Ganze" zu vermeiden und Merge-Konflikte durch klare Absprachen zu minimieren.

Stimmt deine User Storie Schätzung?

Ja. Wir haben als Team mittlerweile ein sehr gutes Gespür dafür entwickelt, welche Anforderungen wie viele Story Points verdienen. Die Schätzungen fühlen sich fundiert und realistisch an.

Hat die Retrospektive einen Gewinn für die Gruppe ermöglicht? Welchen? Wie hast du die Wahl der Retro-Methode empfunden?

Die Methode «Start – Stop – Continue» war solide und hat ihren Zweck erfüllt. Dennoch zeichnet sich eine gewisse Routine ab, da wir diese Form nun schon mehrfach genutzt haben.

Da ich die nächste Retrospektive leiten werde, ist es mein Ziel, eine neue Methode einzuführen. Ein Tapetenwechsel in der Methodik wird helfen, frische Impulse zu setzen und die Aufmerksamkeit im Team hochzuhalten.

Wie hat der Scrum Master heute gearbeitet?

Der Scrum Master hat die Meetings moderiert und geleitet. Abseits der direkten Moderation agierte er eher passiv.

Betrachte die Modulziele auf der ersten Seite dieses Journals. Welche Ziele konntest du bisher erreichen?

Heute konnte ich einen wichtigen Meilenstein erreichen: Ich habe das erste Mal ein Planning geleitet. Damit habe ich nun zu allen Modulzielen bereits praktische Erfahrungen gesammelt. In den kommenden Sprints gilt es nun, diese Kompetenzen weiter zu vertiefen.

Key Learnings

Der Scrum Guide und warum er nur 14 Seiten lange ist

In meiner Auseinandersetzung mit Scrum bin ich auch auf den offiziellen Scrum Guide aufmerksam geworden. Und wow, war ich überrascht: Er ist nur **14 Seiten** lang. Ich dachte, dass ein solches Dokument, welches *die* Methodik eines so ziemlich jeden Software-Teams auf der Welt beschreibt, sicher 500 Seiten lang sein würde. Doch es sind **14 (ja 14!) Seiten**!

Damit kommunizieren die Erfinder von Scrum nicht nur inhaltlich, dass Scrum möchte, dass man sich nicht in stundenlangen Meetings verliert, sondern kommunizieren diese Mentalität auch direkt in der Form, in der sie den Guide an uns hintragen: 14 Seiten kurz.

Ich denke, dass die Scrum-Erfinder möchten, dass man eben nicht viel Zeit mit Planen (= Scrum Guide lesen) verbringt, sondern möglichst schnell in die Umsetzung kommt und Erfahrungen sammelt (= Features shippen!). Scrum ist darauf ausgelegt, dass man sich sofort die Hände dreckig machen soll. Scrum glaubt, dass man eben genau *daraus* die wichtigen Erkenntnisse gewinnt, die das Produkt voran treiben und nicht aus monatelangen Konzeptionsphasen!

Nun: Man darf den Scrum Guide auch nicht unterschätzen. Trotz seiner Kürze ist er sehr tiefgehend: Jedes Wort ist sehr vorsichtig gewählt. Scrum ist keineswegs eine «simple Methode». Sie sieht zwar simple aus, man hat schnell die Grundkonzepte verstanden, aber es ist schwer, extrem gut darin zu werden. Dies ist auch das (inoffizielle) Motto von Scrum:

Simple to understand, difficult to master

Sprint 4 – Donnerstag,19. März + Freitag, 20. März

Wie ist das Planning (Mittwoch) aus deiner Sicht gelaufen?

Ich habe zum ersten Mal ein Planning geleitet. Zuerst habe ich ein paar High-Level-Themen angesprochen, woraus auch ein Task entstanden ist, welcher der PO so sonst nicht auf dem Schirm gehabt hätte. Danach haben wir wie immer die Tasks mit Story Points bevotet. Dies ging ziemlich effizient.

Weniger effizient war aber, dass wir bei vielen Tasks noch viele Details besprechen mussten. Oft war noch kaum etwas beschrieben (der PO schreibt wenn dann nur Stichworte). Wir mussten also die technischen Details jeweils noch ausdiskutieren, damit sich alle im klaren sind, was bei diesem Task denn genau gemacht werden soll (vgl. DoD). Das dauert halt seine Zeit, doch ich denke, dass dies eine wichtige Grundlage für den gesamten Sprint legt. So können die Teammitglieder auch wirklich das tun, was gefragt ist und was der PO will und es geschehen weniger «Abschweifer.»

Key Learnings

Heute nur wenig Story Points gemacht

Ich habe heute nur 1.5 Story Points geschafft. Ehrlich gesagt hat mich das erst mal genervt, weil ich eigentlich den Drive habe, selbst Features zu bauen und im Code voranzukommen. Ich fühle mich irgendwie unproduktiv, wenn ich den ganzen Tag gefühlt „nichts“ geschafft habe, während die anderen fleissig Tickets liefern.

Ich musste mir aber heute klarmachen: Story Points sind keine Zeitmessung. Nur weil ich wenig Punkte auf meinem Namen habe, heisst das nicht, dass ich nicht gearbeitet habe. Als derjenige im Team, der schon etwas besser klar kommt im Code, habe ich heute viel meine Teamkolleg*innen unterstützt und auch einige wichtige Architekturentscheidungen im Team geleitet. Dank diesen auf dem Kanbanboard „unsichtbaren“ Tätigkeiten konnten die anderen heute ungehindert arbeiten und sind gut vorwärtsgekommen.

Das Konzept vom „Enabling“

Heute war meine Hauptaufgabe oft gar nicht das Coden selbst ist, sondern das sogenannte Enabling (habe ich recherchiert). Ich war heute eigentlich die meiste Zeit damit beschäftigt, Fragen zu beantworten oder technische Blockaden bei den anderen zu lösen.

Wenn ich vier Anderen helfe, damit sie ihre 5-Punkte-Tickets fertigbekommen, dann habe ich indirekt auch viel zum Projekt beigetragen, als wenn ich stur mein eigenes Ding durchgezogen hätte. Ohne meine Hilfe wären die anderen vielleicht den halben Tag festgesteckt. Das „Heilende“ an dem Wort Enabling ist, dass man versteht: Mein Wert für das Team ist heute die Unterstützung der anderen gewesen, damit das Projekt als Ganzes läuft.

Das „Big Picture“ und das CSS-Chaos

Was mir heute aufgefallen ist: Wenn jeder von uns nur von Ticket zu Ticket rennt und schaut, dass seine Funktion läuft, geht das Gesamtbild unter. Unsere Web-App sieht z.B. aktuell auf jeder Seite anders aus – andere Abstände, andere Farbtöne etc.

Als Entwickler verliert man im Tunnelblick den Fokus für „die App“ als Ganzes. Ich habe im Planning eingebracht, dass wir mal einen Task haben sollten, um das Design der Seite zu vereinheitlichen.

Anzahl Story Points als KPI?

Ich habe heute über Story Points nachgedacht und recherchiert. Uns wurde gesagt, dass man die nicht nach Zeitaufwand messen soll, und heute habe ich besser verstanden, warum. Ein schwieriges Ticket (hohe Komplexität) kann bei mir schnell gehen, aber bei jemand anderem lange dauern.

Das Problem ist zudem: Es gibt im Team auch einige Dinge, die abseits der Story Points geschehen müssen und dem Produkt sehr helfen.

Ich werde mich nicht mehr schlecht fühlen, wenn meine persönliche Point-Zahl niedrig ist, solange ich sehe, dass das Team wegen mir schneller vorwärtskommt.

Den Abschluss vorbereiten

Der letzte Nachmittag vor der Abgabe war sehr spannend: Wir haben uns erlaubt, statt auf den «main» auf einen «delivery» Branch unsere PRs zu setzen. Dabei habe ich mich angeboten, den delivery Branch zu pflegen. Dadurch konnten wir den Zyklus von Feature-Entwickeln bis auf den Hauptbranch (in diesem Fall «delivery») stark verkürzen. Dadurch konnten wir viel schneller iterieren und noch viele kleine Baustellen, die noch offen waren, lösen.

Ich habe die PRs immer direkt in der IDE ausgeführt und so getestet, ob auch alles funktioniert. Damit konnten wir sehr kurze Zyklen fahren bei gleichzeitig strenger Kontrolle für die Qualität des Produkts.

Rückblickend: Wie erfolgreich konnte das Projekt umgesetzt werden?

Ich bin sehr zufrieden mit dem, was wir erreicht haben. Wir konnten am Schluss leider die Features «Registrierung» und «Produkt anlegen» nicht mehr in den «produktiven» Branch nehmen, da wir dort noch zu viele offene Bugs hatten. Hätten wir noch einen Tag mehr Zeit gehabt, wäre es wohl gegangen. Aber auch ohne diese Features haben wir viel geschafft.

Wenn ihr ohne Projektmethodik gearbeitet hättet: Wäre das schlussendliche Produkt anders herausgekommen? Wie?

Wir hätten wohl dann intuitiv einfach nach Wasserfall gearbeitet. Das Produkt wäre wohl nicht so weit, wie es jetzt ist. Denn der PO brachte immer mal wieder Änderungen und neue Features ein.

Bist du zufrieden mit euerer Arbeit?

Sehr zufrieden. Beim Produkt konnten wir vieles erreichen. Doch letztlich steht für mich der Lernprozess im Vordergrund: Das Ergebnis ist nicht nur unsere Arbeit, sondern auch

Betrachte die Modulziele auf der ersten Seite dieses Journals. Welche Ziele konntest du erreichen?

Ich habe nicht nur die Theorie von Scrum verstanden, sondern auch viel praktische Erfahrung damit machen dürfen. Ich kenne die Begriffe SCRUM, Sprint, Artefakt, BackLog, Daily, Retrospektive und kann diese selbstständig erklären. Ich kenne die Scrum-Rollen und habe auch selbst als Scrum Master während eines zweitägigen Sprints Erfahrungen sammeln können. Zudem konnte ich viele Aufgaben eines «Senior Developer» kennenlernen: Diesen gibt es zwar nicht im Scrum, doch von der Basisausbildung war dies so vorgegeben. Ich bin besser darin geworden, Verantwortung fürs Team und das Produkt zu übernehmen.

Das abschliessende Retro

Dieses Board haben wir währen dem letzten Retro gestaltet. Es zeigt sehr schön, was in der Teamarbeit gut funktioniert hat und welche Dinge uns Schwierigkeiten bereitet hatten.