Kanban im Universitätspraktikum Ein Erfahrungsbericht

Jan Nonnen, Paul Imhoff und Daniel Speicher, Universität Bonn
{nonnen, imhoffj, dsp}@cs.uni-bonn.de

Zusammenfassung

Agile Softwareentwicklung praxisnah und erlebbar zu lehren ist eine Herausforderung, die einen deutlichen Einsatz von Zeit und Personal erfordert. Wir hatten die Möglichkeit, viele gute Erfahrungen in unserer Lehre sammeln zu können. Allerdings hatten wir immer noch einige Schwierigkeiten den Studenten bestimmte Probleme während der Entwicklung bewusst und sichtbar zu machen. In dieser Arbeit berichten wir von unseren Erfahrung mit der Anwendung des Kanban-Entwicklungsprozesses in einem agilen Blockpraktikum an der Universität Bonn. Tatsächlich hat dieser einige dieser Schwierigkeiten deutlicher sichtbar machen können. Basierend auf unseren Erfolgen, bewältigten Herausforderungen und neu beobachteten Schwierigkeiten geben wir abschließend Empfehlungen für andere Lehrende.

Einleitung

Agile Softwareentwicklung ist in den letzten Jahren - mehr noch in der Wirtschaft als in der Lehre - beliebt geworden und hat eine große Verbreitung erfahren (Lindvall u. a., 2004). Dem muss auch die universitäre Softwaretechnik-Lehre Rechnung tragen und sei es nur durch die Befähigung zur kritischen Reflexion. Hier stellt sich aber eine besondere Herausforderung: Agilität möchte der oft erfahrenen Wirklichkeit Rechnung tragen, dass sich Softwareentwicklung nur eingeschränkt planen lässt. Diese Erfahrung und wie sie sich durch agile Vorgehensweisen meistern lässt, ist auf theoretischem Weg kaum vermitteln. Daher haben einige Universitäten spätestens seit 2003 begonnen, Praktika zu agiler Softwareentwicklung in ihre Lehrpläne aufzunehmen (Mügge u. a., 2004; Melnik u. Maurer, 2004). In unseren eigenen Praktika haben wir dabei stets als unerlässlich angesehen, "weiche" Faktoren mit einzubeziehen. So unterstützen wir die Teambildung durch regelmäßige Reflexion und ausdrückliche Retrospektiven, um die Studierenden in die Lage zu versetzen, Kommunikation, Kollaboration und den Prozess als Ganzes zu verbessern.

Ein aktueller Trend der letzten Jahre in der agilen Softwareentwicklung ist *Kanban* (Anderson, 2010). In diesem Beitrag erläutern wir, wie wir Kanban in einem universitären Praktikum zur agilen Softwareentwicklung eingesetzt haben. Wir berichten von unseren Erfahrungen, positiv und negativ, und präsentieren anderen Lehrenden Empfehlungen aufgrund dieser Erfahrungen für ihre Anwendung von Kanban.

1. Visualize Workflow *Visualisiere den Fluss der Arbeit*
2. Limit Work-in-Progress *Begrenze die Menge angefangener Arbeit*
3. Measure and Manage Flow *Miss und steure den Fluss*
4. Make Process Policies Explicit *Mache die Regeln für den Prozess explizit*
5. Use Models to Recognize Improvement Opportunities - *Verwende Modelle um Verbesserungsmöglichkeiten zu identifizieren*

Tabelle 1: Die fünf Kanban-Kerneigenschaften nach Anderson (Anderson, 2010, S. 15).

Wir haben unseren Bericht wie folgt strukturiert: Zuerst geben wir eine allgemeine Einführung in Kanban und berichten über den Praktikumsaufbau und die Historie unserer Praktika. Danach beschreiben wir, wie wir unseren Prozess mit Hilfe von Kanban angepasst haben. Anschließend berichten wir, was wir während des Praktikums geändert haben und am Ende ziehen wir ein Fazit und präsentieren unsere Empfehlungen für Lehrende.

Kanban

Kanban ist ein agiler Software-Entwicklungsprozess, dessen Ursprünge im *Toyota Production System* liegen. Entwickelt wurde das Toyota Production System von Taiichi Ōno, der es auch in seinem 1988 auf Englisch erschienen Buch beschrieb (Ōno, 1988). Das japanische Original war bereits 10 Jahre zuvor erschienen.

"kan" bedeutet wörtlich aus dem Japanischen übersetzt "visuell", und "ban" Karte. Toyota nutzte Kanban, um die Lagerbestände zu reduzieren und einen gleichmäßigen Fluss (Flow) in der Automobilfertigung zu realisieren. David Anderson hat diesen Prozess auf Softwareentwicklung übertragen (Anderson, 2010) und durch fünf Kerneigenschaften charakterisiert (siehe Tabelle 1).

Für Kanban ist die sichtbare Tafel oder ein Whiteboard mit einer Übersicht über den Entwicklungszustand ein zentrales und wohl auch das offensichtlichste Instrument.

Zur Illustration gehen wir schon hier kurz unser eigenes Kanban Board ein: Bereits in der agilen Entwicklung und in unseren vorherigen Praktika war und ist es üblich, die sogenannten *User Stories* auf einem Board zu visualisieren. Dieses Board hatte Spalten für die Entwicklungsschritte wie z.B. ”Analyse” oder ”Done”. In Kanban wird diese Praxis weiter verwendet und angereichert durch die in den fünf Kerneigenschaften genannten Imperative. In Abbildung 1 sieht man ein Bild von einer Kanban Tafel aus dem hier vorgestellten Praktikum. Diese Visualisierung wurde mit Stickern, Whiteboard-Stiften und Klebeband realisiert. Die Verwendung dieser Materialien erlaubte uns, flexibel das Board an das Entwicklerteam und andere Einflüsse anzupassen, und so die Entwicklungsschritte für das Team zu individualisieren. Diese Flexibilität ist wichtig, da Kanban auf die Bedürfnisse zugeschnitten werden sollte und sich diese änderen können. Es sollte daher auch mit dem Team diskutiert werden, was die Phasen (bisher Entwicklungsschritte) sind, durch die die Tickets (bisher User Stories) bis zur Fertigstellung wandern.

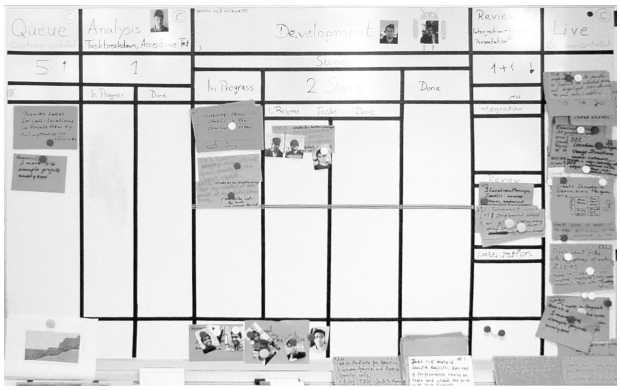

Abbildung 1: Im Praktikum verwendetes Kanban-Board. Das weiter unten referenzierte Video bietet zusätzlich eine dynamische Perspektive auf das Board.

Im Folgenden verwenden wir allgemeine Kanban-Konventionen und orientieren uns an Kanban-Boards welche wir selber eingesetzt und entwickelt haben (siehe Abbildung 2).

Die Stories oder Tickets laufen in der Regel von links nach rechts auf dem Kanban Board und durchlaufen dabei die Entwicklungsphasen bis sie fertig (”done”) sind. Nachdem durch die 2. Kerneigenschaft die Tickets, die in Arbeit sind, begrenzt werden sollten, werden je Spalte Begrenzungen definiert und auf dem Board notiert. Dies bedeutet, dass sobald das Limit in einer Phase erreicht ist, nicht neue Tickets in diese gezogen werden dürfen sondern die Ressourcen genutzt werden müssen um die in Arbeit stehenden Tickets fertig zu stellen.

Kanban basiert auf dem so genannten *Pull-Prinzip*, d.h. dass die Entwickler ein Ticket selbstständig in eine Spalte ziehen, solange die Arbeitslimits in der entsprechenden Phase noch nicht erreicht sind. Sie übernehmen damit die Verantwortung für den der Spalte entsprechenden Schritt in Bezug auf das ”gezogene” Ticket. Des Weiteren werden häufig die einzelnen Phasen in zwei Spalten unterteilt, um zu visualisieren was ”In Arbeit” ist und welche Tickets ”Fertig” sind. Dies erlaubt es in den nachfolgenden Phasen zu sehen, welche Tickets als nächstes gezogen werden dürfen.

Durch die explizite Visualisierung des Ist-Zustandes ist es möglich, den Gesamtprozess zu messen und zu analysieren, wie schnell die einzelnen Tickets durch den Prozess wandern (4. Kerneigenschaft). Ebenso werden Probleme in der Entwicklung sichtbar, wenn man z.B. feststellt, dass die Limits in einer Phase häufig erreicht werden oder die Entwickler in einer Phase regelmäßig auf Arbeit warten müssen.

Diese Informationen können mit genutzt werden, um den Prozess kontinuierlich und gemeinsam zu verbessern (5. Kerneigenschaft). Hier werden häufig auch mathematische Modelle auf Basis der ”Theory of Constraints” (Goldratt u. a., 1984) angewandt um Engpässe (Bottlenecks) in dem Prozess frühzeitig zu erkennen. Diese Modelle geben auch die Möglichkeit, objektiv über den Prozess zu diskutieren. Neben den Limits kann es noch andere Regeln geben die die Bearbeitung von einem Ticket verhindern oder priorisieren (z.B. bestimmte Kartenfarben signalisieren Dringlichkeit).

Damit solche Konventionen nicht zu Fehlern durch Vergessen führen, ist es nützlich, alle Konventionen, die sich im Lauf der Entwicklung ergeben, explizit aufzuschreiben und allen Entwicklern bewusst zu machen (3. Kerneigenschaft). Dieses kann zum Beispiel durch ein Poster mit den Konventionen neben dem Whiteboard geschehen. Dieses fördert analog zu der Verbesserung des Flusses die gemeinsame Diskussion und Verbesserung der Konventionen und Regeln.

Aufbau des Praktikums

Das dieser Arbeit zugrundeliegende Praktikum fand für vier Wochen als Vollzeit-Blockpraktikum im September 2011 während der Vorlesungspause zwischen Sommer- und Winteresemester statt. Blockpraktika dieser Art finden jährlich im Rahmen des *International Program of Excellence in Computer Science (IPEC)* am *Bonn-Aachen International Center for Information Technology (B-IT)* statt, und bieten den Studenten die Möglichkeit Erfahrungen als Softwareentwickler in einem realen Kontext zu sammeln. Seit 2001 werden in den Praktika agile Softwareprozesse gelehrt, angewandt und in Hinblick auf die Lehre angepasst (Mügge u. a., 2004). In der Vergangenheit haben bis zu 16 Studenten in einem Team gearbeitet, welches

dazu von bis zu drei Lehrkräften die ganze Zeit betreut wird.

Dem Praktikum unmittelbar vorgelagert ist ein kleiner Workshop von bis zu drei Tagen. Für diesen vertiefen sich die Studenten in die einzelne Technologien, die im Praktikum genutzt werden, und arbeiten diese in Form von Seminarvorträgen und Ausarbeitungen auf. Neben den Vorträgen wird dieser Workshop vor allem zur Teambildung genutzt. Im Normalfall haben die teilnehmenden Studenten vorher noch nicht zusammengearbeitet. Daher sind teambildende Maßnahmen in der Regel fest in dem Workshop eingeplant. Dieser Workshop hat sich im Laufe der Jahre als nützlich erwiesen, um im Praktikum direkt gemeinsam und produktiv als Team arbeiten zu können.

Der Fokus des Blockpraktikums ist es, einen Einblick in die reale Softwareentwicklung zu geben. Dazu ist das Praktikum als Vollzeit-Präsenzpraktikum ausgelegt, d.h. dass das Team für vier Wochen mit je acht Stunden pro Arbeitstag zusammen arbeitet. Dies hat für die Studenten den Vorteil, das neben dem normalen Arbeitstag keine Überstunden oder Heimarbeiten anfallen. Als eine wichtige Arbeitstechnik hat sich in diesen Praktika das *Pair-Programming* aus dem "eXtreme Programming" (Beck u. Andres, 2004) etabliert. Pair-Programming bedeutet, dass immer zwei Studenten ein Ticket gemeinsam an einem Rechner bearbeiten. Damit sich Wissen zwischen möglichst vielen Teilnehmern gut verteilt, ermutigen wir die Studenten zu regelmäßigem Wechsel des Programmierpartners. Wir halten dafür in einer Matrix fest, wer schon mit wem zusammen entwickelt hat. Dieses Verfahren wurde aus früheren Praktika ohne Änderung übernommen.

Allgemein beinhaltet der tägliche Arbeitstag ein Stand-up Treffen am Morgen und am Abend, sowie ein gemeinsames selbstorganisiertes Teamfrühstück vor Beginn der Arbeit. Wöchentlich gibt es eine Retrospektive (Beck u. Andres, 2004) in der reflektiert wird, was in der Woche erreicht wurde und was man verbessern könnte.

Die drei Lehrkräfte üben während des Praktikums feste aber verschiedene Rollen aus. Wie bereits in (Mügge u. a., 2004) ausgeführt hat sich die Dreiteilung *Kunde*, *Teamleiter* und *technischer Experte* am besten bewährt. Der Kunde legt fest, was er als Produkt entwickelt haben möchte und welche Funktionalitäten er sich wünscht. Das resultierende Produkt kann entweder etwas sein, was in der Forschung genutzt wird oder was von der Industrie an uns herangetragen wird[1]. Der Kunde ist auch in den täglichen Entwicklungsprozess eingebunden wie im Extreme Programming üblich. Das tägliche Projektmanagement des Teams wird von der Rolle des Teamleiters ausgeübt. Dieser stellt die Brücke zwischen dem Entwicklerteam und dem Kunden dar. Die letzte Rolle ist die eines technischen Experten. Dieser ist Ansprechpartner bei technischen Fragen des Teams und hat Wissen in den einzusetzenden Technologien und Projekten.

Anwendung von Kanban

Das Thema des Praktikums in 2011 war die Entwicklung von Analysen für Designprobleme von Java Quellcode für die Android Plattform. Insgesamt nahmen zehn Studenten an dem Praktikum tatsächlich teil, nachdem ursprünglich 16 Plätze vergeben waren. Als Basis gab es schon ein selbstentwickeltes Framework mit dem statische Codeanalysen für Java geschrieben werden konnten. Dieses musste im Rahmen des Praktikums erweitert werden. Ebenso mussten die Studenten erfassen, was hinsichtlich vor allem Performanz und Speicherbedarf problematischer Code auf der Android Plattform ist. Dafür wurden im Laufe des Praktikums z.B. die Entwickler-Richtlinien von Google[2] zu Rate gezogen.

Zur Vorbereitung und Anwendung von Kanban als Entwicklungsprozess wurden die früheren Arbeitsschritte in früheren Praktika analysiert und die Erfahrungen festgehalten. Zu den Bestandteilen früherer Praktika, die wir für erhaltenswert befunden haben gehören (wie bereits beschrieben) das Pair Programming, die Rollenverteilung unter den Mitarbeitern, sowie (wie weiter unten beschrieben) die Differenzierung zwischen Stories und Tasks und ein starker Fokus auf die Selbstorganisation des studentischen Entwicklerteams. Letztere haben wir unter anderem durch die Anleitung zu Stand-Up Treffen und Retrospektiven unterstützt.

Über die Jahre sind wir in den Praktika jedoch auch auf wiederkehrende Schwierigkeiten gestoßen. So ist es immer wieder vorgekommen, dass fast fertige Stories nicht endgültig fertiggestellt wurden. Die letzten integrierenden Arbeitsschritte wurden von den Studenten sowohl als unerfreulich als auch als technisch anspruchsvoll wahrgenommen. Daher wurden andere Arbeiten der Fertigstellung vorgezogen, auch wenn sich das Team bereits dieses Problems bewusst war. Sicherlich hätte sich dieses Problem durch Zuordnung einzelner zu diesen Arbeitsschritten lösen lassen, aber das hätte dann unser Bemühen das Team in seiner Selbstorganisation zu unterstützen deutlich konterkariert. Mit Kanban erhofften wir uns, dass durch die Kombination des Pull-Prinzips mit strikten Limits für die einzelnen Phasen, die Sichtbarkeit des Problems so deutlich würde, dass sich das Team seiner annehmen würde.

Eine weitere Herausforderung, die von den Studenten deutlicher wahrgenommen wurde als von uns, war den Aufwand für Besprechungen - insbesondere zum Anfang einer Iteration - zu reduzieren. In nicht wenigen Praktika verbrachten die Studenten einen Tag oder sogar etwas mehr damit, in die Produktwünsche des Kunden eingeführt zu werden. Bei einer

[1] Ein Beispiel für ein entwickeltes Produkt: `http://www3.uni-bonn.de/Pressemitteilungen/299-2009`

[2] `http://developer.android.com/guide/practices/performance.html`

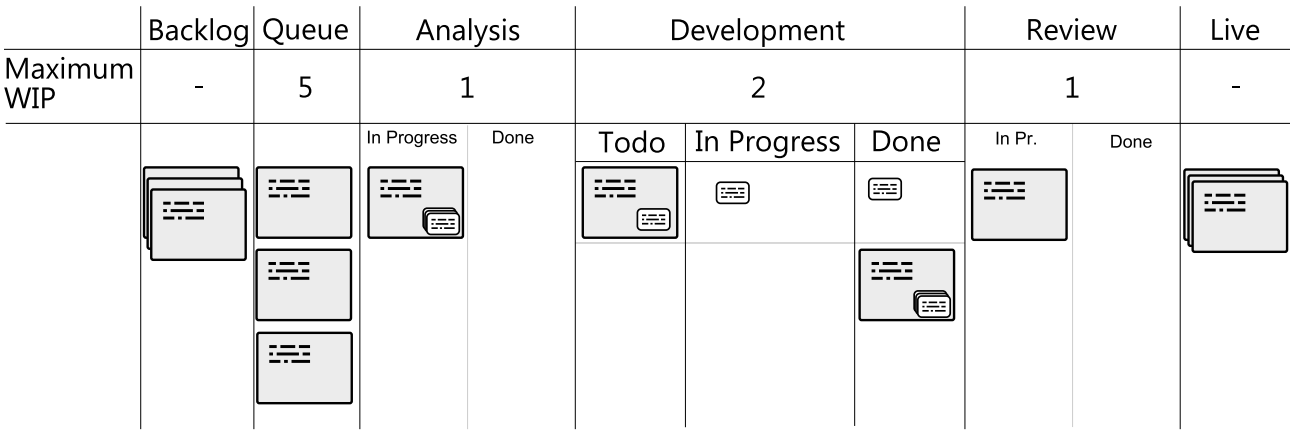

Abbildung 2: Schematische Darstellung des Kanban-Boards des Praktikums mit den einzelnen Phasen und Limits.

Teamgröße von mindestens 10 Studenten führt dies zumindest vorübergehend zu einiger Passivität. Da Kanban weniger Wert auf Iterationen legt bzw. sogar ganz ohne diese auskäme, versprachen wir uns, die initialen Besprechungen etwas entzerren zu können.

Das Ende der Iterationen hingegen stellte den Kunden vor die Herausforderung, Funktionalitäten in einem Umfang überprüfen zu müssen, den er nicht bewältigen konnte. Daher war das Feedback des Kunden über die Qualität des Geleisteten mitunter eher unscharf. Durch einen gleichmäßigeren Fluss fertiggestellter Funktionalität hofften wir dieses Problem entschärfen zu können. Dieses konnten wir schließlich durch vorhergehende Reviews durch Team und Teamleiter beheben.

Basierend auf den Erfahrungen bisheriger Praktika wurde der grundsätzliche Arbeitsfluss in einem initialen Kanban-Board festgehalten. Das initiale Board enthielt die folgenden sechs Phasen: Backlog, Input Queue, Analysis, Development, Review und Live (Abbildung 2).

Das Backlog enthält die Ideen des Kunden, die in noch nicht ausreichend genauer Form definiert sind oder die (noch) nicht realisierbar sind. Dies entspricht dem Backlog des Scrum-Prozesses mit dem Unterschied, dass die Tickets noch nicht von dem Team geschätzt wurden und sie noch in Bearbeitung sind. Die ”Input Queue” Phase wird von dem Kunden kontrolliert und bestimmt, welches das nächste zu entwickelnde Feature ist.

In der Analyse wird mit dem Kunden zusammen festgelegt, was alles zu der Realisierung des Tickets gehört, d.h. es werden Akzeptanzkriterien festgelegt. Außerdem nimmt das Team in der Analyse eine Zerlegung des Tickets in kleinere sogenannte Tasks vor. Diese Unterscheidung von Tickets, die einen durch den Kunden wahrnehmbaren Funktionalitätsfortschritt haben, (User Stories) und den dazu nötigen Arbeitsschritten (Tasks) haben wir aus den vorhergehenden Praktika übernommen. Dadurch wird es möglich einerseits die Teilschritte genau zu beschreiben und auf unterschiedliche Entwicklerpaare zu verteilen und andererseits bleibt der angestrebte Fortschritt sichtbar und verschwindet nicht hinter technischen Details. Die Implementation findet in der Development-Phase statt. Anschließend werden sowohl ein Team-interner Codereview, als auch ein gemeinsamer Review mit dem Kunden durchgeführt. Nach erfolgreichem Abschluss der Reviews ist ein Ticket in der Live-Phase angelangt.

Zur Visualisierung der Phasen auf einem Whiteboard wurde je Phase eine Spalte verwendet (siehe Abbildung 2). Die Phasen Analyse, Development und Review wurden noch weiter unterteilt in die Zustände ”In Progress” sowie ”Done”. Die Arbeitslimits wurden in einer separaten Zeile je Phase dargestellt. Ab der zweiten Praktikumswoche wurde ein Zeitraffervideo des Kanban-Boards erstellt[3]. In diesem sieht man, wie einzelne Tickets von links nach rechts durch die einzelnen Phasen des Prozesses wandern.

Wie erwähnt unterteilt das Team die anfänglichen Tickets des Kunden (die User Stories) in kleinere Tasks, die von den Studenten jeweils in einem Paar in einer kurzen Zeit bearbeitet werden können. In der Entwicklungsphase wandern daher die Task-Karten für ein Ticket von links nach rechts. Ein Ticket ist dann fertig implementiert, wenn alle Tasks des Tickets fertig implementiert wurden. Da es aber Tickets geben kann mit wenigen oder auch welche mit vielen Tasks, gibt es für diese Tasks keine Beschränkung hinsichtlich maximal erlaubter Anzahl an Tasks, die zeitgleich bearbeitet werden dürfen. Hingegen wurde die Anzahl an Tickets - also der User Stories zu den Tasks - in der Development-Phase beschränkt.

Während des Praktikums wurde der aktuelle Zustand zu festen Zeiten von allen Tickets auf dem Kanban Board gemessen und festgehalten. Aus diesen Daten wurde anschließend ein ”Cumulative-Flow” Diagramm (Anderson, 2010) (siehe Abbildung 3) generiert. In dem Diagramm wird über die Zeit aufgetragen, wieviele Tickets sich zu einem Zeitpunkt

[3] http://www.youtube.com/watch?v=0iPaIefQbgM

im System befinden und welchen Zustand diese haben. Dabei zählen fertig bearbeitete Tickets zu der Live-Phase. Eine breite Phase in dem Diagramm kann entweder bedeuteten, dass ein Ticket lange in dieser Phase bearbeitet wird, oder dass die nächste Phase nicht verfügbar ist und die Tickets nicht weiterbearbeitet werden können. Dieses Diagramm wurde im Praktikumsraum neben dem Board aufgehangen und diente als Visualisierung des Fortschritts in täglichen Stand-Up Besprechungen mit dem Team.

In Kanban-Prozessen findet man häufiger spezialisierte Entwicklerrollen (z.B. Architekten oder Tester) die in speziellen Phasen arbeiten. In unseren Praktika legen wir Wert darauf, dass die Studenten alle Entwicklungsphasen erfahren, so dass wir keine spezialisierten Rollen genutzt haben. Als Konsequenz daraus waren die Studenten frei in der Wahl in welchem Entwicklungsschritt sie arbeiten möchten, solange die Kanban-Limits berücksichtigt wurden. Allerdings gab es Priorisierungen von zu bearbeitenden Tickets im System.

Initial war die einzige Regelung, dass Tickets in späteren Phasen bevorzugt weiterbearbeitet werden sollten, damit der Kunde zügiger fertige Features bekommt. Auch wenn diese Regel bereits dem Toyota Production System entstammt, war sie für uns besonders plausibel, da in früheren Praktika - wie oben erwähnt - die Integration von Tasks zum Ticket und der Review unbeliebter waren. In der Konsequenz wurde lieber ein neues Ticket analysiert als ein fast fertiges Ticket fertigzustellen und dem Kunden zu präsentieren.

Anpassungen während des Praktikums

Während des Praktikums wurden einige Anpassungen an den Arbeitsfluss und die Visualisierung des Prozesses vorgenommen. Im Folgenden möchten wir diese genauer beschreiben und vorstellen. Der Hauptgrund für die Anpassungen war ein ”verebben” des Ticketflusses innerhalb des Prozesses. Ebenso wurden Veränderungen aus didaktischen und allgemeinen praktischen Gründen vorgenommen.

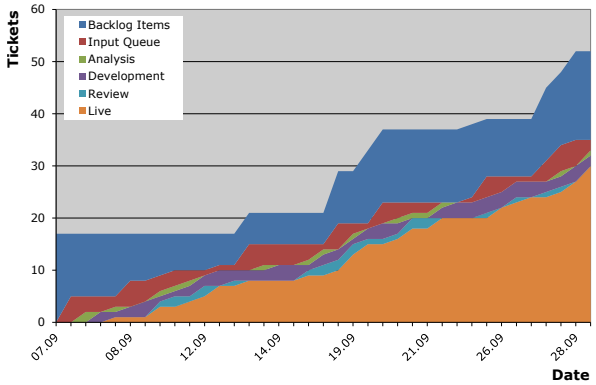

Abbildung 3: ”Cumulative-Flow” Diagramm des Praktikums

Das Verebben des Ticketflusses wurde deutlich anhand der Anzahl der Entwickler die keine Arbeit hatten. Dies wurde teilweise durch blockierende Tickets in späteren Phasen erzeugt. Im Kanban-Prozess, sollten idealerweise die freien Ressourcen mithelfen die Engpässe zu beheben. Dies war aber bei unseren feingranularen Tasks die schon auf ein Entwickler Paar geplant waren selten möglich. Ebenso zeigte sich, dass unsere oben beschriebene anfängliche Priorisierung ungünstig gewählt war. Durch die Engpässe in der Development-Phase, gab es wenige bis keine Tickets für den Review. Nachdem aber auch der Review priorisiert werden sollte, wurde eher der Review gemacht bevor ein neues Ticket in die Analyse-Phase gezogen wurde.

Es stellte sich auch heraus, dass ein Ticket in der Analyse-Phase nicht von mehreren Paaren effektiv bearbeitet werden konnte. Daraus resultierte ein Stocken des Entwicklungsprozesses, in dem keine Tickets in der Analyse oder fertig für die Development-Phase waren. Als Lösungsmöglichkeiten hätten wir die Arbeitslimits in der Development-Phase anpassen können um ein Entwicklerpaar frei zu haben für Analyse oder Review. Wir passten aber unsere unglücklich gewählte Priorisierung an, indem wir die Priorisierung Analyse -> Review -> Development vorschlugen. Dies stellte sich im Laufe des Praktikums als sinnvolle Entscheidung heraus, da es den Engpass entlastete und die Studenten in der Development-Phase Tickets zum Bearbeiten hatten.

Ein weiterer Engpass in dem Prozess war die Rolle des Kunden. Dieser wurde in der Analyse-Phase zur Akzeptierung der genaueren Anforderungen und in dem finalen Schritt der Review-Phase benötigt. Da es nur einen Kunden gab, war dieser eine begrenzte Ressource. Ebenso war es aber weder sinnvoll, die Anzahl an Kunden zu erhöhen noch die Einbindung des Kunden in beide Schritte zu vermeiden, daher wurde versucht, durch den Teamleiter in beiden Schritten einen vorgelagerten Kundenrepräsentanten zu spielen. Dies bewirkte, dass viele Fragen der Studenten in diesen Schritten von dem Teamleiter beantworten werden konnten und der Kunde trotz alledem die fertigen Dokumente präsentiert bekam und selber entscheiden konnte, ob sie fertig sind.

Für die meisten Studenten war es das erste Mal, dass sie in einem größeren Entwicklerteam an einem gemeinsamen Projekt arbeiten mussten. Der Kanban-Prozess war keinem der Studenten vor dem Praktikum bekannt. Eine Einführung in den Kanban-Prozess und das Kanban-Board war zwar durch in den Workshop gegeben, aber diese war nach unserer jetzigen Erfahrung zu kurz. Aus diesen beiden Tatsachen resultierten während des Praktikums Unsicherheiten der Studenten, was zu tun ist oder wo sie helfen konnten. Nachdem die Erfahrung mit dem Prozess und dem

Board merklich zunahm, wurden die Zeiten, in denen die Studenten aufgrund der Limits keine Arbeit am Projekt leisten konnten, für die Studenten demotivierend.

In der Kanban-Literatur (Anderson, 2010) wird dieser nützliche Spielraum ("Slack") genutzt, um einerseits den Prozess zu optimieren und andererseits den Entwicklern die Möglichkeit zu bieten, sich selbstständig fortzubilden. Als Fortbildungsmöglichkeit definierten wir einige Möglichkeiten in Form von Tutorials und Literatur, die sich um die benutzte Technologie oder solche, die später noch genutzt werden sollte, drehten. Diese hatten damit den zusätzlichen Nutzen für den Kunden, dass das Wissen über die Technologien frühzeitig im Team verteilt werden konnte. Die Studenten hatten bei Leerlauf die freie Wahl aus diesen auf Karten beschriebenen Fortbildungen zu wählen.

Neben diesen Karten konnten die Studenten auch jederzeit Vorschläge oder Ideen einbringen, wie man den Gesamtprozess oder das Board verbessern könnte. Ein Vorschlag war zum Beispiel, dass nicht immer klar war welche Studenten an welchen Tickets arbeiteten. Nachdem die Tickets in der Development-Phase in meist mehrere Tasks unterteilt und mit je einem Paar weiterentwickelt wurden, war es wichtig, dass sich diese Studenten über den Zustand austauschen, da es häufig Abhängigkeiten zwischen den Tasks gab. In diesen Schritten ist es daher wichtig, zu wissen, wer an welchen Tasks arbeitet um sich z.B. räumlich nebeneinander zu setzen. Als Lösung des Problems wurde ein Bild von jedem Entwickler und Dozenten gemacht und auf dem Board befestigt. Jeder hatte dann die Aufgabe, sein Bild auf das Ticket oder den Bereich zu verschieben, an dem er arbeitete. Da es diese Bilder auch für Teamleiter und Kunden gab, half diese einfache Idee als Nebeneffekt auch, den Paaren, die in der Analyse oder dem Review arbeiteten, visuell zu erfassen ob der Kunde verfügbar war.

Fazit

Um zu bewerten, wie gut unser Prozess im Praktikum funktionierte, wurde das Cumulative Flow Diagramm (siehe Abbildung 3) herangezogen. Außerdem hatten wir an einem Tag während des Praktikums einen externen Prozess-Reviewer, der Entwickler-Teams in der Industrie bei der Einführung von Kanban professionell berät. Von ihm erhielten wir hilfreiches Feedback und wertvolle Tipps. Schließlich haben wir zudem die Studenten noch mit einem Fragebogen um ihre Einschätzung gebeten.

In unserer Nachbereitung stellten wir aber fest, dass es für uns schwer war zu entscheiden, ob uns das Diagramm eine Aussage über die Prozessqualität machen kann. Dies hängt vor allem daran, dass selten ein festes Paar ein Ticket durch den gesamten Prozess begleitet hat. Daher hängt die Dauer, die ein Ticket in einer Phase verbracht hat, nicht nur von den Eigenschaften des Tickets selber ab, sondern auch von den Entwicklern und der Kommunikation zwischen den Phasen. Durch die kurze Zeit des Praktikums ist es darum schwer aus den wenigen Messpunkten herzuleiten, ob Kanban rein objektiv zur Verbesserung des Gesamtprozesses beigetragen hat.

Als weiterer Kritikpunkt fügt unsere finale Priorisierung von Analyse > Review > Development dem Gesamtprozess noch eine zusätzliche Komplexität hinzu. Diese Entscheidung sollte auf der einen Seite bewirken, dass dem Kunden möglichst zügig Tickets als fertig vorgelegt werden konnten. Auf der anderen Seite sollte es durch die Priorisierung immer fertig analysierte Tickets geben, die direkt in den Development-Schritt gezogen werden konnten. Diese beiden Schritte waren gegeben, trotzdem waren wir mit der resultierenden Priorisierung nicht glücklich, da es eine höhere Aufmerksamkeit auf Seiten der Studenten forderte.

Durch den ungewohnten Entwicklungsprozess und das Umfeld waren die Studenten anfangs eher passiv in dem Prozess. Mit steigender Erfahrung mit Kanban und steigendem Selbstbewusstsein im Team fingen die Studenten an, Verbesserungsvorschläge zu machen. Durch eine bessere und längere Einführung von Kanban in dem Workshop könnte man diese Hemmschwelle vermutlich verringern und frühere Verbesserungsvorschläge erleichtern. Wie so oft in unseren Praktika war das Team in der letzten Woche erst auf voller Produktivität.

Eine Schwierigkeit in der Anpassung des Prozesses während des Praktikums ist der kurze Zeithorizont. Man kann entweder zur Optimierung Arbeitsschritte und Puffer hinzufügen oder entfernen, oder an den Limits in jedem Schritt drehen. Wenn man eines von beiden anpasst, soll man laut Literatur (Anderson, 2010) einige Zeit warten, bis sich der Prozess auf die neuen Rahmenbedingungen eingependelt hat.[4] Ebenso sollte man nur eine Schraube in jedem Schritt verändern. Dies bedeutete aber für unser Praktikum, dass es schwer war den Effekt abzuschätzen. Wir hatten uns daher auch entschieden während des Praktikums keine Arbeitsschritte anzupassen und nur die Limits der Schritte zu justieren. Davon haben wir auch in der ersten Woche Gebrauch gemacht, indem wir die Limits in der Development-Phase von sechs Tasks auf zwei Tickets angepasst haben.

Zur Evaluation des Praktikums haben wir die Studenten am Ende gebeten, einen Fragebogen auszufüllen. Neben acht offenen Fragen enthielt der Fragebogen Fragen, die mit Werten auf einer Skala von 0 bis 6 zu beantworten waren. Unter anderem baten wir die Studenten um ihre subjektive Einschätzung ihrer Kompetenz vor und nach dem Praktikum. Hier bedeutete 0 "keinerlei Wissen" und 6 "exzellentes Wissen". In allen abgefragten Kompetenzen verbesserten sich

[4] Kanban-Entwickler aus der Wirtschaft berichteten uns im persönlichen Gespräch, dass man bis zu einem Monat warten muss, um zu sehen ob eine Änderungen positive Auswirkungen hatte

die Studenten im Mittel um mindestens 1,1 Punkte. Den größten mittleren Zuwachs gaben die Studenten bei den technischen Fertigkeiten an (Verständnis von Android (+1,7), Prolog (+1,9), Verwendung von Subversion (+2,0) und Eclipse (+2,1)). Aber auch bei "weichen" Faktoren war der Zuwachs deutlich (Kommunikation (+1,5), Zusammenarbeit (+2,0)). Auch im Hinblick auf Kanban äußerten sich die Studenten eindeutig positiv. So hat ihrer Meinung nach das Kanban Board die Übersicht verbessert (5,0) und den Arbeitsfluss klar widergespiegelt (5,1). Zudem waren die Work-in-Progress Limits klar (4,7). Nicht ganz so zufrieden waren die Studierenden mit der eigenen Handhabung von "Staus" (3,0).[5]

Empfehlungen für die Lehre

Die Evaluation ergab, dass der Kanban-Entwicklungsprozess für die Studenten nützlich war. Außerdem, hat der Prozess Probleme verringert, die wir in früheren Praktika erlebt haben. Zum Beispiel hatten wir erlebt, dass blockierende Tickets früher nicht immer allen sichtbar und bewusst waren und zusätzlich von dem Prozess nicht explizit behandelt wurden. Zu diesem Zweck hat, das Kanban-Whiteboard in dem Praktikumsraum eine essentielle Rolle als Mittel für Diskussionen geboten. Dieses hat auch den Studenten geholfen den Gesamtüberblick zu behalten und ihnen die Möglichkeit gegeben selbst ihre eigenen Leistungen zu reflektieren. Wir würden daher, trotz digitalen Möglichkeiten, empfehlen, ein physisches Kanban-Board zu nutzen und es zentral in den täglichen Arbeitsalltag zu stellen.

In Scrum (Schwaber u. Beedle, 2001) gibt es feste Iterationen, die im vorhinein geplant werden. Dem Kunden ist es dabei nicht möglich, den Fokus der Entwicklungs dynamisch während einer Iteration anzupassen. Durch die flexible Input-Queue in unserem Praktikum war es dem Kunden jederzeit möglich den aktuellen Fokus anzupassen. Dies ist in einem Forschungsprojekt wichtig, da auf aktuelle Erkenntnisse und Entwicklungen reagiert werden kann.

Die Fortbildungsmöglichkeiten für Studenten während des Lehrlaufes waren unserer Meinung nach sinnvoll, wurden aber nicht gut genug kommuniziert. Die Arbeit an diesen Aufgaben entsprach auch nicht dem allgemeinen Ticketfluss. Wenn man solche Möglichkeiten vorsieht, sollte man diese Aufgaben auch an einem Ort in der Nähe des Whiteboards visualisieren. Der Vorteil davon ist, dass sowohl die Lehrenden als auch die Teilnehmer wissen an welchen Aufgaben die Studenten arbeiten.

Tägliche "feature celebrations" nach der Mittagspause wurden genutzt um im Team kurz vorher fertiggestellte Tickets in einer Livedemonstration zu feiern. Dies hatte den Zweck, den Studenten bewusst zu machen, was sie bereits erreicht haben. Nachdem die Studenten häufig nicht in allen Phasen bei jedem Ticket involviert waren, erzeugte dieses Zelebrieren häufig auch das Gefühl, dass jeder seinen Teil dazu beigetragen hat und der Code vom gesamten Team stammt. Wir haben diese Treffen auch genutzt, um über die Entwicklung des Tickets selber zu reflektieren: Was lief gut und was hätte im Prozess besser laufen können?

Das Praktikum hat drei Probleme aufgezeigt, denen man sich bewusst sein sollte. Erstens, sollte vor einem Praktikum genug Zeit investiert werden um den Prozess und Kanban zu besprechen und zu diskutieren. Zusätzlich sollten die Studenten die Möglichkeit bekommen, den Prozess selber zu erleben, zum Beispiel in Form eines Spieles.

Zweitens waren die unterschiedliche Größen und Komplexitäten der Tickets ein Problem. In Kanban sollten idealerweise alle Tickets ungefähr gleich groß sein und lange brauchen. Durch gemeinsame Schätzungen der Tickets durch das Team konnten im späteren Teil des Praktikums zu große Tickets identifiziert werden. Zu Anfang des Praktikums fehlen im Normalfall die Erfahrungen zu präzisen Schätzungen. Dozenten sollten in dieser Phase verstärkt dem Team mit ihrer Erfahrung helfen und Wert darauf legen, ihr Wissen über Schätzungen an das Team zu vermitteln.

Drittens war die Anpassung der Arbeitsschritte oder der Limits eine Herausforderung. Anpassungen sollten in kleinen Schritten erfolgen und man sollte einige Zeit abwarten, ob die Änderungen den gewünschten Effekt haben. In unseren Praktika haben wir aber nur eine kurze Zeit zur Verfügung, und die Studenten benötigen unserer Erfahrung nach einige Zeit um sich an die Rahmenbedingungen des Teamentwicklung und der Technologien zu gewöhnen. Wenn man den Prozess einsetzen will, sollte man sich als Lehrender dessen bewusst sein und in der Vorplanung bereits berücksichtigen. Ebenso empfehlen wir in so einer kurzen Zeitspanne entweder sich vorher zu überlegen ob die Arbeitsschritte verändert werden dürfen, oder die Limits in den Arbeitsschritten angepasst werden sollen. Wir hatten uns in dem Praktikum entschieden nur die Limits anzupassen und hatten damit gute Erfahrungen gesammelt.

Praktikum 2012

Aufgrund unserer guten Erfahrungen wurde auch im Praktikum im Jahr 2012 der Kanban-Prozess eingesetzt. Als Basis wurde das hier vorgestellte Kanban-Board genutzt. Dieses musste nicht weiter angepasst werden, so dass die Dozenten eine geringere Vorbereitungszeit benötigten. Im Laufe des Praktikums wurden auch nur wenige und erwartete Anpassungen an den Limits der Prozessschritte vorgenommen. Diese Anpassungen wurden nötig, da die problematische Priorisierung des vorherigen Praktikums entfernt wurde. Weiterhin mit Problemen behaftet waren wiederum

[5] Die Fragen und die Verteilung der Antworten können online nachgelesen werden: `http://sewiki.iai.uni-bonn.de/_media/teaching/labs/xp/2011b/agile_lab_2011_evaluation.pdf`

unterschiedlich komplexe Tickets und ihr Einfluss und Aussagekraft auf den Gesamtfluss. Durch die Arbeitslimits bedingt konnten aber auch schwierige Aufgaben fokussierter bearbeitet werden.

Zusammenfassung

In dieser Arbeit haben wir von unseren Erfahrung mit der Anwendung des Kanban-Entwicklungsprozesses in einem Blockpraktikum an der Universität Bonn berichtet. Neben den Vorteilen des Prozesses und den Lösungen früherer Probleme wurden auch neue Probleme aufgezeigt. In einem kürzlich durchgeführten Praktikum konnten diese teilweise auch schon bewältigt werden.

Als wichtigste Erkenntnis würden wir jedem Lehrenden, der Kanban einsetzen möchte, empfehlen, zu Anfang den bisher eingesetzten Prozess und dessen Schwierigkeiten schriftlich festzuhalten. Auf Basis dessen kann man dann überlegen ob Kanban sinnvoll ist und wie man den Prozess anpassen kann. In unserem Fall konnte Kanban die hartnäckigen Schwierigkeiten aufgeschobener Fertigstellung fast fertiger Stories und ungenügender Reviews aufgrund auf das Iterationsende konzentrierter Fertigstellungen aus der Welt schaffen, sowie den Besprechungsaufwand deutlich reduzieren.

Danksagung

Wir möchten uns bei Matthias Bohlen für seinen externen Review unseres Kanban-Prozesses während des Praktikums bedanken. Wir haben wertvolle Tipps erhalten, die unser Verständnis von Kanban vertieft und unseren Prozess verbessert haben.

Literatur

[Anderson 2010] ANDERSON, D.: *Kanban, Successful Evolutionary Change For Your Technology*. Blue Hole Press, 2010

[Beck u. Andres 2004] BECK, K. ; ANDRES, C.: *Extreme programming explained: embrace change*. Addison-Wesley Professional, 2004

[Goldratt u. a. 1984] GOLDRATT, E.M. ; COX, J. ; OUTPUT, C.: *The goal: Excellence in manufacturing*. Bd. 262. North River Press New York, 1984

[Lindvall u. a. 2004] LINDVALL, M. ; MUTHIG, D. ; DAGNINO, A. ; WALLIN, C. ; STUPPERICH, M. ; KIEFER, D. ; MAY, J. ; KAHKONEN, T.: Agile software development in large organizations. In: *Computer* 37 (2004), dec., Nr. 12, S. 26 – 34

[Melnik u. Maurer 2004] MELNIK, G. ; MAURER, F.: Introducing agile methods: three years of experience. In: *Euromicro Conference, 2004. Proceedings. 30th*, 2004, S. 334 – 341

[Mügge u. a. 2004] MÜGGE, H. ; SPEICHER, D ; KNIESEL, G.: Extreme Programming in der Informatik-Lehre - Ein Erfahrungsbericht. In: *Informatik 2004 - Beiträge der 34. Jahrestagung der GI, Lecture Notes in Informatics*, 2004

[Ōno 1988] ŌNO, T.: *Toyota production system: beyond large-scale production*. Productivity Pr, 1988

[Schwaber u. Beedle 2001] SCHWABER, K. ; BEEDLE, M.: *Agile software development with Scrum*. Bd. 18. Prentice Hall, 2001