Agilität hautnah

Die Woche eines agilen Entwicklers – Teil 1: Montag bis Mittwoch

Alex Bepple, Henning Wolf

Agile Entwicklung bedeutet Entwickeln zwischen Testen, inkrementellem Design, Architekturdiskussionen, Aufwandsschätzungen, Fertigstellungsprognosen und Abstimmungen im Team. Der Artikel beschreibt aus der Sicht der Entwickler, welche Herausforderungen auf sie zukommen: Wie agil entwickelt wird; wie sich das Design ergibt; wer, wann und wo und wie über Architekturen diskutiert; wie die Qualitätssicherung erfolgt und sich alles immer noch mehr zum Guten wenden soll. Was haben Entwickler davon? Wirklich weniger Stress und Hektik? Steht nicht ständig der nächste Termin an?

Klar hat man schon von agilen Methoden gehört, manches machen die meisten längst agil. Aber wie sieht es denn nun konkret aus, wenn die Projektmanagementmethode Scrum heißt oder nach eXtreme Programming oder Feature Driven Development entwickelt wird? Führt agile Softwareentwicklung zu besserer Qualität und hoher Wartbarkeit? Oder ist hohe Qualität und gute Wartbarkeit Voraussetzung für agile Softwareentwicklung?

Im Folgenden stellen wir das fiktive Tagebuch eines Entwicklers vor. Im ersten Teil (Montag bis Mittwoch) geht es um das Projektsetting, die Arbeitsumgebung, agile Architekturen und Architekturdiskussionen sowie die Planung und die Aufwandsabschätzung. Im zweiten Teil (Donnerstag bis Freitag) betrachten wir Testerfahrungen, Code-Ownership und Retrospektiven.

Montagmorgen: Das Setting

Seit ein paar Monaten fahre ich nicht mehr ins Büro meines Arbeitgebers, sondern direkt zu unserem Kunden. Dort habe ich mit meinen Entwicklerkollegen seit Projektbeginn meinen Arbeitsplatz und direkt gegenüber auf dem Flur sitzt unser Hauptansprechpartner, Herr A. Das ist ziemlich praktisch, wenn man mal eine Frage hat. Das scheint dem Kunden auch so zu gehen, denn er kommt gar nicht so selten mit Fragen zu uns. Oft beziehen die sich dann auf zukünftige Features. Das ist OK, kann aber auch mal zu viel werden, sodass wir eine Zeit lang schon „störungsfreie Zeiten“ vereinbart hatten: jeweils vormittags und nachmittags zwei Stunden am Stück ohne wechselseitige Unterbrechungen.

Wir sind fünf Entwickler im Team und sitzen in einem Raum, der in der Größe nur knapp für uns reicht. Der Raum hat drei Schreibtische und wir sitzen jeweils zu zweit an einem Schreibtisch zum Programmieren in Paaren, also dem gemeinsamen Entwickeln. Wir könnten auch mehr Platz haben – wir haben noch einen Nebenraum –, aber den nutzen wir nur selten für Besprechungen oder so. Da stehen auch noch zwei Rechner, wenn wir mal nicht in Pairs arbeiten oder jeder seine E-Mail bearbeitet.

Anfangs dachte ich, dass ich den Krach und die Nähe nicht lange aushalte. Mittlerweile kann ich mir gar nicht vorstellen, anders zu arbeiten. Und wirklich laut ist es äußerst selten. Es ist sogar sehr nützlich, dass man manchmal in Gesprächsfetzen der anderen Entwickler mitbekommt, was sie gerade tun. Dann kann man schnell seinen Senf dazugeben und kommt gemeinsam zu besseren Lösungen. Man kann auch sehr schnell Probleme diskutieren und Fragen stellen. Sonst würde man öfter 15 bis 30 Minuten im Code nach Antworten suchen, statt in den Nebenraum zu gehen; jetzt fragt man schnell, und irgendwer weiß es meist.

Die Wände unseres Raumes sind vollgeklebt mit Flipcharts und Haftnotizen (s. Abb. 1). Auf den Flipcharts sind die Ergebnisse unserer letzten Architekturdiskussionen, auf den Haftnotizen stehen die Aufgaben der aktuellen Iteration. Sie sind dabei so angeordnet, dass man auf einen Blick erkennen kann, wie viel wir schon geschafft haben und was noch offen ist.

Auf zwei Flipcharts stehen unsere Teamregeln, also Vereinbarungen, die wir im Team gemeinsam getroffen haben. Beispielsweise wollen wir konsequent testgetriebene Ent-

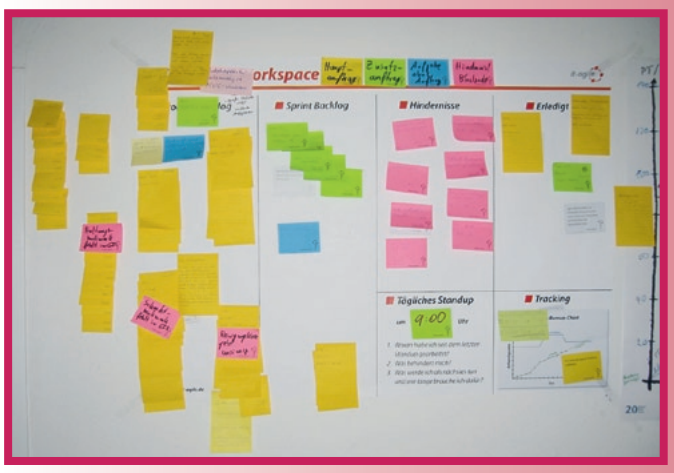

Abb. 1: Informative Workspace

wicklung betreiben. Wir haben aber auch eine Vereinbarung, dass wir um 9 Uhr gemeinsam anfangen. Denn wir machen alles im Team, es gibt keine abgeschotteten Verantwortungsbereiche. Der Code wird von allen geändert, Entscheidungen werden von allen mitgetragen. Da müssen wir schlicht auch zusammen anfangen. Wer schon früher da ist, bearbeitet noch seine E-Mails oder fängt vielleicht auch schon einmal mit einer Aufgabe an. Aber große Unterschiede gibt es da nicht. Eine andere Vereinbarung im Team lautet nämlich, dass wir mindestens bis um 17 Uhr da sind.

Ich bin heute etwas knapp dran, aber gerade noch rechtzeitig zum Standup-Meeting um 9. Wir beantworten reihum drei Fragen: Was habe ich seit dem letzten Standup getan? Was werde ich bis zum nächsten Standup tun? Was hat mich behindert? Weil ich zu spät dran war, kann ich die zweite Frage heute noch nicht beantworten, denn ich habe am Freitagnachmittag meine Aufgabe zusammen mit meinem Pair-Partner abgeschlossen und mir bisher keine neue Karte genommen. Da aber Torben noch an einer Aufgabe arbeitet und wir laut Pair-Plan heute gemeinsam arbeiten, ist es nicht so schlimm. Es ist also klar, was wir heute zu Ende machen, bevor wir uns eine neue Aufgabe nehmen. Nach dem Standup gehe ich noch kurz zu Hans, weil ich genauer wissen will, wie er den neuen Auftragsstatus integriert hat. Das war nämlich bei meiner letzten Beschäftigung mit dem Thema im Code eine echte Herausforderung.

Torben erklärt mir, womit er und Marcus sich am Freitag beschäftigt haben und was noch zur Erledigung fehlt. Ich habe noch eine Idee, wie wir etwas von ihrem Freitagscode ein wenig vereinfachen können, und wir erledigen gemeinsam die Aufgabe. Danach bringen wir den Aufgabenzettel unter „erledigt“ an der Wand an. Das heißt: Wir haben alles kompiliert, alle Tests laufen lassen und noch einen manuellen Akzeptanztest durchgeführt. Gut, dass wir automatische Tests haben. Früher war viel manuelles Rumklicken nötig, um zumindest eine geringe Sicherheit zu haben, dass wir keine unerwünschten Seiteneffekte produziert haben. Die Benutzerhilfe haben wir auch gleich aktualisiert, damit sie gar nicht erst anfängt zu degenerieren.

Zur neuen Aufgabe haben wir sofort eine fachliche Nachfrage und gehen gemeinsam zu unserem Kundenansprechpartner, der uns den Sachverhalt erläutert. Wir notieren uns die Details und gehen wieder in den Team-Raum. Kurz erklären wir die fachlichen Besonderheiten auch unseren Kollegen. Dann machen wir uns an die Aufgabe. Manchmal unterbrechen wir unsere Kollegen nicht im Arbeitsfluss und geben die Informationen des Kunden im nächsten Standup an alle weiter.

Dienstag: Architekturdiskussion

Heute bin ich eine Viertelstunde früher da und beantworte noch zwei E-Mails. Im Standup kommt bei Hans und Marcus ein technisches Problem hoch. Man soll nämlich zukünftig Aufträge auf Mobilgeräte synchronisieren können; da müssen wir uns ausdenken, was das für unser Zustandsmodell bedeutet, weil die auf die Mobilgeräte synchronisierten Aufträge auf dem zentralen System dann nicht veränderlich sein sollen. Natürlich kann man das Problem nicht in einem Standup lösen, und das ist ja auch nicht die Idee der Standups. Stattdessen haben wir für den Vormittag einen einstündigen Termin zur Diskussion vereinbart. Wir vereinbaren immer solche festen Zeitrahmen, um in den Besprechungen fokussiert nach einer Lösung zu suchen und vorher zu wissen, wie lange wir von unserer sonstigen Arbeit abgehalten werden. Wenn mal eine Stunde nicht reicht, vereinbaren wir am Ende noch ein Meeting, dann allerdings häufig in kleinerem Kreis oder mit externem Beistand, z.B. einem erfahrenen Architekten oder auch mal einem Technologieexperten – je nach Problemstellung.

Hans und Marcus haben das Meeting schon etwas vorbereitet, sodass sie uns das Problem sehr genau vorstellen können. Sie haben zwei Lösungsmöglichkeiten auf Flipcharts aufgezeichnet, und wir diskutieren nun die Vor- und Nachteile. In der Diskussion entsteht noch eine dritte Variante. Allerdings merken wir auch, dass wir eine entscheidende fachliche Frage nicht beantworten können: Es macht in den Modellen einen ziemlichen Unterschied, ob die Aufträge maximal für einen Tag auf ein Mobilgerät synchronisiert werden können, also abends immer wieder zurücksynchronisiert werden, oder ob diese Dauer beliebig sein kann. Gut, dass der Kundenansprechpartner so nah ist. Wir erläutern ihm die fachliche Konstellation, und er versichert uns, dass es immer, aber auch wirklich immer, abends wieder zurücksynchronisiert wird. Wir unterhalten uns noch kurz über den Fehlerfall, wenn es trotzdem einfach nicht passiert, und vereinbaren, dass dann der Auftrag am nächsten Tag einfach wie nie heraussynchronisiert betrachtet werden darf. Gut, dass wir nachfragen konnten, denn es führt zu einer noch einfacheren Lösung, die wir nach knapp einer Stunde auf einem Flipchart haben und im Raum an der Wand befestigen.

Danach geht die Entwicklung weiter. Heute bin ich wegen der ungeraden Anzahl an Entwicklern alleine. Ich wähle deswegen aber keine andere Aufgabe, sondern werde den neuen Code einfach morgen früh meinem Pair-Partner vorstellen. Ich verzichte auch alleine nicht darauf, den Code regelmäßig einzuchecken. Mein Pair-Partner morgen hat Gelegenheit, mir Feedback zu geben, und wir können bei Bedarf noch gemeinsam etwas ändern. Das Feedback ist kein formales Review, aber es erhöht die Sicherheit, dass ich allein nichts Wesentliches vergessen habe, unser Code im Team sehr ähnlich aussieht und wir unsere gemeinsamen Standards einhalten.

Mittwoch: Planungsmeeting, Aufwandsschätzung

Heute reviewen wir gemeinsam mit dem Kunden den aktuellen Projektstand. Das machen wir jeden Mittwoch so. In der Regel sind alle Stakeholder des Kunden repräsentiert: Herr A, der Projekt-Koordinator auf Kundenseite ist, die Fachanwender, ein IT-Verantwortlicher, der IT-Support. Das gesamte Entwicklungsteam nimmt teil. Unser Projektleiter ist auch dabei, wenn er vor Ort ist. Gestern haben wir die Agenda vorbereitet: Welche Probleme gilt es anzusprechen? Welche ganz oder teilweise entwickelten Anforderungen können wir vorführen und uns Feedback einholen?

Letzte Woche haben wir noch Klärungsbedarf zu einer Anforderung festgestellt: Die beauftragte Umsetzung birgt Probleme im Mehrbenutzerbetrieb. Dies ist nicht bedacht worden. Nun wollen wir gemeinsam mit dem Kunden nach einer Lösung suchen. Zu einer anderen Anforderung hat es beim letzten Review harsches Feedback von unserem Ansprechpartner gegeben: Er möchte mehr Funktionalität sehen. Nach unserer Meinung fordert er mehr als beauftragt. Auf Nachfragen hin ist er sich aber auch nicht mehr ganz sicher, welche Lösung im Sinne der Anwender ist. In den verschiedenen Organisationseinheiten variieren die Abläufe teilweise. Um für beide Anforderungen Lösungen zu finden, die allen dienen, haben wir zu dem dieswöchigen Review zusätzlich Fachanwender eingeladen.

Für den IT-Verantwortlichen des Kunden ist heute nur wenig Interessantes dabei. Er verlässt das Meeting nach kurzer Zeit

wieder. Unser Projektleiter fehlt ganz. Aber das behindert niemanden: das Entwicklungsteam kann auch allein laufen.

Gemeinsam mit Herrn A und den Fachanwendern klären wir beide Anforderungen. Alle bringen Ideen ein: die Kundenseite genau so wie wir selbst. Uns fällt auch die Rolle zu, auf Schwierigkeiten im Ablauf oder nicht bedachte Wechselwirkungen mit existierenden Funktionen im System hinzuweisen. Nach einigem Hin und Her kristallisieren sich zwei praktikable Lösungen für das erste Problem heraus. Für die zweite Anforderung wünschen sich die Fachanwender die gleiche Zusatzfunktionalität wie Herr A.

Für beides ist weniger Aufwand veranschlagt worden, als die jetzt entwickelten Lösungen erfordern. Wir stellen uns die Aufgabe, die Lösungen genau zu schätzen und dann mit dem Kunden zu verhandeln, wie viel er zusätzlich beauftragen muss, damit wir die von ihm gewünschte Funktionalität umsetzen. Herr A versucht uns gegenüber gerne, seine Wünsche nach zusätzlicher Funktionalität als Fehler im System auszulegen. Mit ihm darüber zu verhandeln, fällt auch in die Verantwortung des Teams. Wir stellen fest, dass uns der Verhandlungsspielraum für die beiden Anforderungen, um die es geht, nicht klar ist, und beschließen, hierzu Rücksprache mit unserem Projektleiter zu halten.

Herr A hat noch mehr Gepäck zum Meeting mitgebracht. Für eine bereits beauftragte Zusatzanforderung wird mehr Aufwand nötig. Das Budget ist hierbei kein Problem. Wir weisen ihn aber darauf hin, dass sich durch den zusätzlichen Aufwand möglicherweise der nächste Release-Termin für den Hauptauftrag verschieben wird.

Außerdem möchte Herr A wissen, wie teuer es wäre, zwei weitere Zusatzanforderungen umzusetzen. Wir sagen ihm zu, die Schätzungen einzuplanen und ihm die Ergebnisse dann mitzuteilen.

Früher haben wir uns sofort an solche Schätzungen gemacht und damit unsere Entwicklungsgeschwindigkeit teilweise spürbar verringert. Seit einigen Monaten haben wir ein anderes Vorgehen. Wir repräsentieren solche Schätzaufgaben durch andersfarbige Aufgabenkarten. Pro Iteration planen wir nur noch eine begrenzte Anzahl davon ein. So stellen wir sicher, dass wir stets mit dem Hauptauftrag vorankommen. Sollten sich solche Aufgabenkarten stauen, haben wir einen Hinweis auf ein Durchsatzproblem und können dies gemeinsam mit unserem Projektleiter lösen.

Nach dem Review des Projektstands planen wir unsere Iteration. Diese dauert bei uns eine Woche. Erst schauen wir uns an, was wir in der letzten Woche geschafft haben, und tracken unsere Aufwände. Wir haben in der letzten Woche weniger Aufgaben geschafft, als wir uns vorgenommen hatten. Wir machen dafür hauptsächlich Probleme bei der Integration mit einem Drittsystem verantwortlich und merken uns vor, bei der nächsten Retrospektive zu diskutieren, wie wir diese Schnittstelle in Zukunft effizienter managen können.

Nun planen wir die Aufgaben für die nächste Woche ein. Dazu wählen wir aus allen noch zu erledigenden Aufgaben einzelne aus und hängen sie gleich in den Bereich an unserer Wand, der die aktuelle Iteration repräsentiert. Wir planen ein, eine Zusatzanforderung sowie eine der problematischen Anforderungen zu schätzen, über die wir im Meeting gesprochen haben. Dann füllen wir die Iteration mit Entwicklungsaufgaben für den Hauptauftrag auf.

Im nächsten Teil finden Sie die Tagebucheinträge für Donnerstag und Freitag sowie Anmerkungen zu Testerfahrungen, Code-Ownership und Retrospektiven.

Alex Bepple ist Softwareentwickler bei der akquinet it-agile GmbH in Hamburg. Er ist ScrumMaster und überzeugter testgetriebener Entwickler. Aktuell arbeitet er in einem Scrum-/XP-Team. E-Mail: alex.bepple@it-agile.de.

Dipl.-Inform. **Henning Wolf** ist Geschäftsführer der akquinet it-agile GmbH. Er war jahrelang extremer Programmierer. Heute berät er Kunden bei der Einführung agiler Methoden. Er ist Koautor des 2008 bei dpunkt erschienenen Buchs „Agile Softwareentwicklung". E-Mail: henning.wolf@it-agile.de.