Alle Infos zur JAX ››› ab S. 39

S&S

entwickler magazin

entwickler magazin

www.entwickler-magazin.de

März/April 2.2013

Mac & iOS

- So (un)sicher ist Mac OS
- Teamwork: Xcode und CocoaPods

Xtext

Die Effizienz-maschine im Fi...

...quests ...richtig handeln

Graphen-datenbanken

Fünf gute Gründe für den Einsatz

Das Netz der zwei Geschwindigkeiten

Die wichtigsten HTML5-APIs

SAP Research analysiert Nachhaltigkeit des ASE-Trainings

Gekommen, um zu bleiben

Bis Ende des Jahres 2012 haben ca. 3 000 Entwickler bei SAP das Programm „Agile Software Engineering“ (ASE) durchlaufen. Dieses Programm hat SAP gemeinsam mit andrena objects konzipiert. Als Ergänzung zu Scrum vermittelt es agile Engineering-Techniken und -Praktiken. So unterstützt es Entwicklungsteams im eigenen Bestreben, dauerhaft Software hoher Qualität zu produzieren. 2010 wurde mit den ASE-Schulungen begonnen.

von Dr. Jürgen Heymann

Dreitausend Mal jeweils vier Wochen Training – bei einer solchen Investition stellt sich die Frage, ob sich dieser Aufwand auszahlt. Und die Antwort sollte präziser sein als ein ungefähres Stimmungsbild. Deshalb befragte SAP Research ca. 1 000 Entwicklerinnen und Entwickler, inwieweit sie agile Techniken wie die Testgetriebene Entwicklung und das Pair Programming adaptiert haben und welche Erfolge sie beobachten. Außerdem untersuchte SAP Research, was die Entwickler heute, etwa ein Jahr nach dem Training, von agilen Methoden halten. Das Ergebnis: eine insgesamt hohe Zielerfüllung, die jetzt in klaren Zahlen dokumentiert ist. Besonders auffällig: 75 Prozent der Befragten sagen, dass agile Praktiken die Qualität signifikant verbessern, und das laut 59 Prozent davon ohne Geschwindigkeitsverlust oder sogar mit Gewinn.

Was den Aufwand von vier Wochen Training für 3 000 Entwicklerinnen und Entwickler rechtfertigt, ist das große Ziel der ASE-Schulung: Sie soll die Softwarequalität dauerhaft steigern, während der Umfang (Scope) mindestens gleich bleibt. Auch Verbesserungen hinsichtlich der Teamarbeit zählen zu den Erfolgen, die das ASE-Training anstrebt.

Konzipiert wurde das Training von SAP und andrena objects aus der wichtigen Lehre heraus, dass die bloße Einführung von Scrum als Rahmenwerk zur Projektsteuerung oft nicht den gewünschten Erfolg brachte. Weder stieg die Produktivität in allen Fällen wie erwartet, noch steigerte sich die Softwarequalität quasi automatisch. Offensichtlich reicht es nicht aus, ein Intervall für die Produktion bestimmter Software vorzugeben, solange unklar bleibt, wie die Teams sie überhaupt produzieren können. Damit rückt wieder in den Mittelpunkt, was in der IT-Welt streckenweise als unbedeutendster Teil des Softwareprojekts angesehen wurde: das reine Programmieren oder, um seinem tatsächlichen Anspruch gerechter zu werden, das Software-Engineering. Dieses kreative „Handwerk“ bildet den Schwerpunkt des Trainings, das konsequenterweise Agile Software Engineering heißt und sich grundsätzlich an das ganze Scrum-Team richtet. Denn Programmieren ist hier Teamsache.

Während also die Motivation klar ist, mit der SAP und andrena objects das ASE-Programm gestartet haben, so blieb doch eine entscheidende Frage bisher offen, nämlich die, in welchem Maß das Training das tägliche Arbeitsverhalten ändert.

Nachhaltigkeitsstudie

Die Antwort darauf liefert die Nachhaltigkeitsstudie von SAP Research. Sie dokumentiert in konkreten Zahlen, wie es etwa ein Jahr nach dem Training um die Anwendung der agilen Methoden steht. Eine Frage, die umso interessanter ist, als die Teams grundsätzlich selbst entscheiden, ob und welche agilen Techniken sie adaptieren. Jeder Zwang zur Übernahme agiler Ar-

beitsweisen ist bei SAP ausgeschlossen. Schließlich sind Selbstorganisation und Selbstbestimmung konkrete Ziele der Agilität.

Aber sollten die Techniken aus einer überzeugenden, auf jahrelanger Erfahrung beruhenden Schulung nicht auf jeden Fall in reale Projekte einfließen – zumal ein Programm wie ASE schon so konzeptioniert ist, dass man sich bereits während des Trainings hinaus aus den Laborbedingungen und hinein in die Praxis begibt? Eine Woche steht für die Theorie zur Verfügung, danach begleitet der Trainer das Team für drei Sprints in reale Projekte.

Warum SAP Research die Ergebnisse der Nachhaltigkeitsstudie trotzdem als zumindest teilweise offen ansah, dafür gibt es einen einfachen Grund, der in der Natur der agilen Methoden liegt: Die angestrebte Steigerung der Softwarequalität steht, der ASE-Überzeugung nach, in direkter Relation zur Agilität der Entwicklung. Agilität ist dabei mehr als eine Mode oder oberflächliche Kosmetik, sie ist auch mehr als nur eine Änderung der Arbeitsorganisation. Agilität steht für einen tiefgreifenden Wandel der Arbeitstechniken und fordert damit oft einen mächtigen Gegner heraus: die pure Gewohnheit. Das ist auch der Grund, warum die agilen Methoden auf ihrem Weg von der Schulung in die Praxis in der Versenkung verschwinden können. Viele dieser Versenkungen tragen bekannte Namen. Sie heißen „übliche Vorgehensweise“ oder „bewährter Prozess“, werden mit Überzeugungen wie „Never touch a running system“ ausgehoben oder mit Sätzen aufgerissen, die mit „Ja, aber“ anfangen und letztendlich einfach „nein“ oder „Das will ich nicht“ meinen.

Viele der SAP-Teams wollen. Das belegt die Nachhaltigkeitsstudie. So würden mehr als drei Viertel der Befragten in ihrem eigenen Unternehmen agile Praktiken einsetzen, um nur eines der Ergebnisse vorwegzunehmen.

Befragt wurden etwa 1 000 Entwicklerinnen und Entwickler. SAP Research lud alle im Jahr 2011 ASE-geschulten Teams ein, an der Studie teilzunehmen. Etwa ein Jahr nach dem Training sollten die ehemals Teilnehmenden angeben, welche agilen Methoden sie übernommen haben und welche Auswirkungen sie beobachten, beispielsweise im Hinblick auf die Menge an Software, die per Sprint geliefert wird oder die Qualität des erzeugten Codes. Antworten kamen auch von Product Ownern, jener Gruppe, die innerhalb des Scrum-Teams die „andere Seite“ vertritt und besonders kritisch urteilen dürfte, was Entwicklungsgeschwindigkeit und Umfang anbelangt.

Die Ergebnisse im Einzelnen

Das dokumentierte Ergebnis: Bereits die Hälfte der Teams erreicht beim Pair Programming und bei der Testgetriebenen Entwicklung die Zielgröße. Im Idealfall wird etwa 50–60 Prozent der Zeit mit einem Partner programmiert, der aktuelle Durchschnittswert liegt der Umfrage zufolge bei 36 Prozent. Ähnlich sieht es bei der Testgetriebenen Entwicklung aus: 37 Prozent des neuen Codes entsteht im Schnitt mit dieser Technik, als Ziel gelten 60–70 Prozent. Auch hier erreicht ca. die Hälfte der Teams diesen Zielwert. Die hohe Zielerreichung ist deswegen so interessant, weil Testgetriebene Entwicklung (TDD) und Pair Programming, das Programmieren mit einem Partner, als Methoden gelten, die, konsequent angewendet, die größten Erfolge in Richtung Agilität bringen. Dennoch ist hier das Feld geteilt: Viele Entwicklerinnen und Entwickler finden beide Methoden sehr angenehm und auch emotional positiv. Andere jedoch sind skeptisch. In der Studie bezeichneten 34 Prozent der Befragten TDD als ASE-Praktik, die mit dem größten Stress verbunden ist, 23 Prozent stuften Pair Programming als anstrengend ein. Das ebenfalls agilitätstypische Refactoring dagegen sorgte nur bei 6 Prozent der Befragten für Stress.

Dass gerade TDD und Pair Programming als aufreibend empfunden werden können, ist nicht überraschend. Immerhin dokumentiert sich der tiefgreifende Wandel der Arbeitsgewohnheiten gerade in diesen beiden Techniken. Pair Programming beispielsweise bildet den diametralen Gegensatz zum Ideal des „einsamen Wolfes“. Auffällig ist, dass die teaminterne Adaption des Pair Programming für knapp die Hälfte der Teams kohärent ist. In diesen Teams bejahen also nahezu alle Entwicklerinnen und Entwickler das Programmieren mit einem Partner. Insgesamt lehnen nur etwa 15 Prozent das gemeinsame Programmieren komplett ab. Dagegen zählt die Hälfte der Teams zu den „High Adoptern“: Sie arbeiten durchschnittlich 61 Prozent der Entwicklungszeit im Pair-Programming-Modus und liegen damit schon am oberen Ende der Optimalwerte.

Für diese „High Adopter“ erfüllen sich nach eigener Einschätzung die größten Versprechen, die agile Methoden wie Pair Programming machen: die deutliche Steigerung der Softwarequalität und die Verbesserung der Teamarbeit. Auch die befürchtete Verlangsamung beim gemeinsamen Programmieren bleibt nicht nur aus, teilweise berichten die Befragten im Gegenteil sogar, in der gleichen Zeit mehr Features liefern zu können. Nicht zu unterschätzen ist ebenfalls der Wissenstransfer innerhalb des Teams, der das gesamte Know-how steigert. Außerdem halten die „High Adopters“ den Code für deutlich verständlicher als vor der Einführung des Pair Programming. Offenbar sorgt die bei dieser Methode elementare Kommunikation unter den Entwicklerinnen und Entwicklern für ein klares Plus an gemeinsamem Wissen und Transparenz.

Das Gleiche gilt für die „High Adopters“ des Test-driven Developments. Auch sie berichten von einer höheren Softwarequalität und erleben die Teamarbeit als verbessert. Selbst beim gelieferten Umfang verzeichnen sie ein moderates Plus. TDD scheint jedoch – im Gegensatz zum Pair Programming – in gewisser Hinsicht eine individuelle Praktik zu sein und eher keine Teamfrage. In einigen Teams schwankte der Grad der TDD-Adaption zwischen 0 und 100 Prozent. Die „High Adopter“ entwickeln durchschnittlich 61 Prozent der Zeit testge-

trieben. Damit erreichen sie den als ideal eingestuften Wert.

Da TDD eine hohe Testabdeckung mit sich bringt, erzielen die überzeugtesten Anwender eine durchschnittliche Testabdeckung von 88 Prozent – was als optimal gilt. Auch beim Refactoring werden erfreuliche Zahlen gemeldet: In der idealen agilen Entwicklerwelt fließen 20-25 Prozent der Zeit ins Refactoring, der Durchschnitt in der Umfrage liegt bei 23 Prozent.

Die Gruppe der Product Owner schließlich bestätigt die Einschätzung der Entwicklerinnen und Entwickler: Auch die „andere Seite“ sieht eindeutige Verbesserungen in der Teamarbeit und in der Qualität der Software, während, wie erhofft, die Zahl der per Sprint gemeldeten Defekte sank. Der bei agilen Methoden häufig befürchtete Zeitverlust bleibt der Umfrage zufolge aus.

Fazit und Ausblick

Zusammenfassend belegt die Umfrage, dass sich die agilen Methoden schon gut etabliert haben – und vor allem, dass sie die versprochenen Erfolge wirklich bringen. Ein Anlass, die Hände in den Schoß zu legen, ist das allerdings nicht. Stattdessen formuliert SAP jetzt konkrete Maßnahmen, um die schleichende Aushöhlung der ASE-Prinzipien zu verhindern. Für die SAP-interne ASE-Initiative hat es hohe Priorität, die Kommunikation über Agile Software Engineering lebendig zu halten. Angedacht sind dazu etwa Anwendertreffen, Vorträge und Blogs, die Themen aus den Trainings wiederholen, aber auch ganz neue Aspekte aufgreifen. Außerdem sollen diejenigen, die sich an die agile Methodik halten, besondere Anerkennung genießen. Kontinuierliches Lernen und Nachhaltigkeit, das sind zwei wichtige Ziele der Initiative. Dabei sorgen die Ergebnisse der Umfrage und erste Anwenderstudien für besondere Motivation. Denn sie belegen, dass die agilen Praktiken den Teams wirklich dabei helfen, kontinuierlich Software hoher Qualität zu entwickeln – auch wenn es bedeutet, eingeschliffene Gewohnheiten komplett zu ändern. Parallel zum Maßnahmenkatalog laufen die Schulungen in Kooperation von SAP und andrena objects weiter.

2013 sollen ca. weitere 1 500 Entwickler ein ASE-Training erhalten. Darüber hinaus wird ein Angebot der Kurse für Kunden vorbereitet.

Dr. Jürgen Heymann hat Informatik in Braunschweig, Atlanta und München studiert. Er ist Chief Development Expert bei SAP in der Abteilung „CPO Software Engineering“ und leitet dort das ASE-Programm.

Notizen:

