Agile Software Engineering

Alles kann besser werden

Höhere Produktivität und Qualität in Softwareprojekten – dafür wurde für die SAP AG das Training Agile Software Engineering (ASE) konzipiert. Inzwischen gibt es das Training auch für die Microsoft-Technologien.

Auf einen Blick

Stephan Dahm arbeitete zunächst als Softwareentwickler in Java und C# bei der andrena objects ag. In diesem Rahmen konnte er in den letzten fünf Jahren in diversen Projekten in Scrum und eXtreme Programming Erfahrung sammeln. Heute ist er unter anderem als Coach und Trainer für ASE tätig.

Jochen Winzen ist Senior Softwareentwickler und Coach für agile Methoden (Scrum, ASE) bei der andrena objects ag. Nach mehreren Jahren Programmierung mit Java und Migrationsprojekten mit Microsoft .NET (C#) übernimmt er in seinen aktuellen Projekten die Rollen Requirements Engineer und Product Owner. Zudem leitet er das Geschäftsfeld Karlsruhe II (Team .NET).

Inhalt

dnpCode

A1301ASE

Das Vorgehensmodell Scrum liefert allgemein anerkannte Praktiken zur Projektplanung, zum Controlling und zur Teamorganisation. Offen lässt es jedoch Fragen nach der Wurzel des Ganzen, dem Software Engineering und vor allem nach der konkreten Engineering-Praxis.

Kurz zusammengefasst könnte man sagen, dass Scrum ein Regelwerk ist, das ein ausbalanciertes Kräftegleichgewicht zwischen dem Product Owner und dem Development Team herstellt. Dabei fungiert der Scrum Master als Wächter der Balance.

Nun lautet eines der großen Versprechen von Scrum Just-in-time-Projektsteuerung und -Erfolgskontrolle, eine direkte Reaktion auf Gegebenheiten also, agil eben. Erfolgskontrolle setzt entsprechend klar formulierte (Test-)Anforderungen voraus. Und es setzt voraus, dass die entsprechenden Werkzeuge und Best Practices zur Verfügung stehen, um beispielsweise innerhalb der vorgegebenen Zeitspanne das geforderte Softwareteilstück auch tatsächlich erstellen zu können.

Damit erscheint es sinnvoll, in Scrum eine Vorgehensweise zu integrieren, die auch diejenigen Sphären eines Softwareprojekts detailliert beleuchtet, die den Code beziehungsweise die Entwicklungsprinzipien betreffen. Genau hier setzt das von SAP (www.sap.com) und andrena objects (www.andrena.de) gemeinsam entwickelte „Agile Software Engineering“ (ASE) an – ein Trainingsprogramm für Scrum-Teams.

Ein Projekt. Vier Sphären.

Das ASE-Programm gliedert jedes Projekt im Software Engineering in vier Sphären, die als konzentrische Kreise eine Hülle bilden um das Zentrum, den Code.

Projektplanung und -steuerung: Die erste, äußerste Hülle beschreibt die Einbettung des Projekts in das Unternehmen und seine Geschäftsprozesse. Unter dem Titel Projektplanung und -steuerung markiert sie das Aufeinandertreffen der direkt am Projekt Beteiligten – des Product Owners, des Development Teams und des Scrum Masters – auf die nur indirekt am Projekt beteiligten Stakeholder, die Anforderungen an die fertige Software stellen. Sie ist gekennzeichnet durch die Auslieferung von Softwareteilstücken (Inkrementen) innerhalb vorgegebener Zeitintervalle. Nach der Scrum-Nomenklatur heißen diese Iterationen Sprint.

Organisation des Development Teams: Die zweite Sphäre beschreibt die Organisation des Development Teams. Gemäß den Scrum-Grundsätzen übernehmen die Self-empowered Teams die Hoheit über die Planung ihrer Aufgaben und deren Umsetzung.

Allerdings liefert das reine Scrum keine Handlungsanweisungen dazu, welche Arbeitstechniken und Werkzeuge das selbstorganisierte Team nutzen kann, um am Ende jedes Sprints tatsächlich ein Teilstück verwendbarer Software liefern zu können.

Engineering-Praktiken: Daraus erklärt sich unmittelbar die Bedeutung der Engineering-Praktiken in der dritten Sphäre. Sie beantwortet die Frage, wie die grundsätzliche Forderung nach höherer Qualität und Produktivität konkret erfüllbar wird. Dazu stellt sie die testgetriebene Entwicklung in den Mittelpunkt. Zentrale Praktiken sind Unit Testing, Refactoring, Pair Programming und Continuous Integration.

Code: Im Zentrum des ganzen Projekts schließlich steht der Code selbst. Er entsteht nach den Engineering-Prinzipien der objektorientierten Entwicklung, die vor 40 Jahren festgelegt wurden und deren fundierte Kenntnis beim reinen Scrum als Voraussetzung gilt.

Anders ausgedrückt leitet sich das ASE-Programm aus der Überzeugung ab, dass jedes erfolgreiche Software-Engineering-Projekt drei Dimensionen hat – die Projektsteuerung, das Software Engineering und, als Treiber, ein detailliertes Requirements-Management.

Dazu ergänzt das ASE-Programm den Scrum-Kanon an Best Practices um Arbeitstechniken aus eXtreme Programming, Codierungs- und Entwurfsmustern und professioneller Projektpraxis. ASE versucht also nicht, etwas ganz Neues zu erfinden.

Das ASE-Programm integriert sich in Scrum, etabliert verlässliche Software-Engineering-Praktiken und legt so das Fundament, auf dem Scrum seine Prinzipien und das damit verbundene Potenzial voll entfalten kann. Das wirklich Neue am ASE-Programm in der Projektsteuerung ist die bewusste Anerkennung des

Software Engineerings als Grundvoraussetzung für den Erfolg des Development Teams.

Das Trainingsprogramm

ASE verfolgt drei Ziele:

- Die Produktivität und Qualität zu erhöhen, indem professionelles Software Engineering konsequent angewandt wird.
- Mehr Planungssicherheit zu schaffen für das Entwicklungsteam und den Product Owner.
- Komplexe Projekte besser steuerbar zu machen; dazu werden einsetzbare Softwareteilstücke im Takt von Scrum Sprints ausgeliefert.

Diese drei Ziele machen unmittelbar deutlich, warum sich das ASE-Training an ganze Teams und nicht an Einzelne richtet: Alleine die Forderung nach mehr Planungssicherheit kann nur erfüllt werden, wenn innerhalb des gesamten Teams – Product Owner, Development Team und Scrum Master – maximale Transparenz herrscht. Außerdem ist die Stärkung von Teams als handelnde und entscheidungsfähige Einheiten ein zentrales Prinzip des Vorgehensmodells Scrum.

Das Programm ASE wurde von verschiedenen Software-Ingenieuren von andrena und SAP aus ihren Praxiserfahrungen abgeleitet. Eine Lehre aus diesen Erfahrungen war, dass ein Training unter Laborbedingungen nur bedingt aussagekräftig ist für die Übertragbarkeit des Erlernten in die Realität.

Daher wurde ASE als Zwei-Stufen-Programm etabliert, in dem auf die erste Stufe, die Schulung unter Laborbedingungen, das Coaching und Training direkt im Projekt folgt. Die erste Stufe dauert eine Woche und dient dazu, die zentralen Arbeitstechniken unter Reinraumbedingungen zu vermitteln, etwa die Definition of Done (DoD), Refactoring oder Backlog Estimation Sessions.

In der zweiten Stufe werden die ASE-Trainer direkt im realen Projekt aktiv. In drei einwöchigen Sprints setzen Trainer und Team die erlernten Techniken weiter um. Schätzen (Estimation), Planen (Sprint Planning), inkrementelle Lieferung (Incremental Delivery) und klar definierte Qualitätsstandards (Definition of Done) – das sind die Eckpunkte des Scrum-Prozesses.

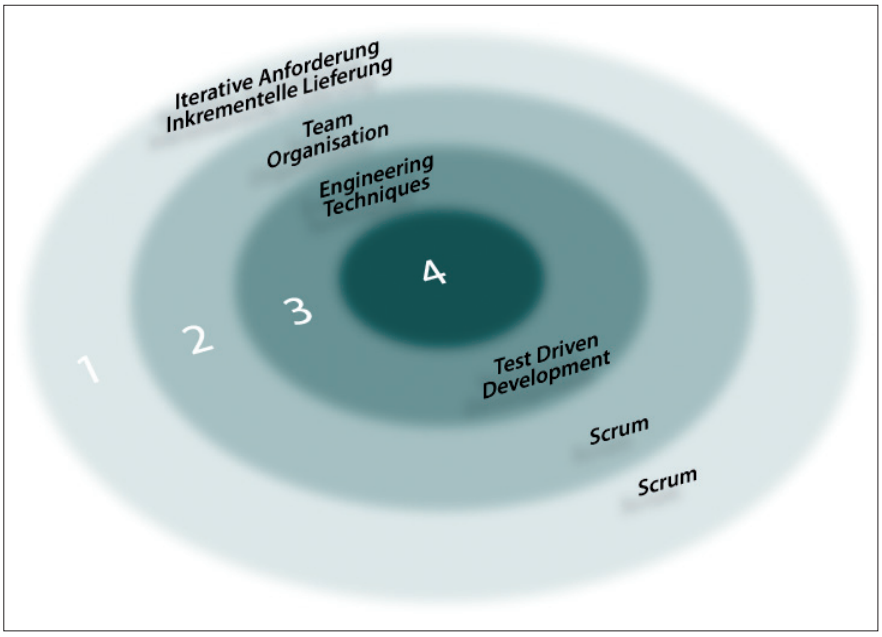
Mit jedem absolvierten Sprint und mit jedem ausgelieferten Software-Inkrement steigt die Genauigkeit, mit der das Team Zeit, Umfang und Qualität abschätzen kann. Gleichzeitig wächst das Vermögen, im vorgegebenen Intervall tatsächlich ein fertiges, sprich einsetzbares Software-Inkrement zu liefern.

Das ASE-Programm etabliert einen Prozess, mit dem die Schwarmintelligenz des Teams kontinuierlich zunimmt.

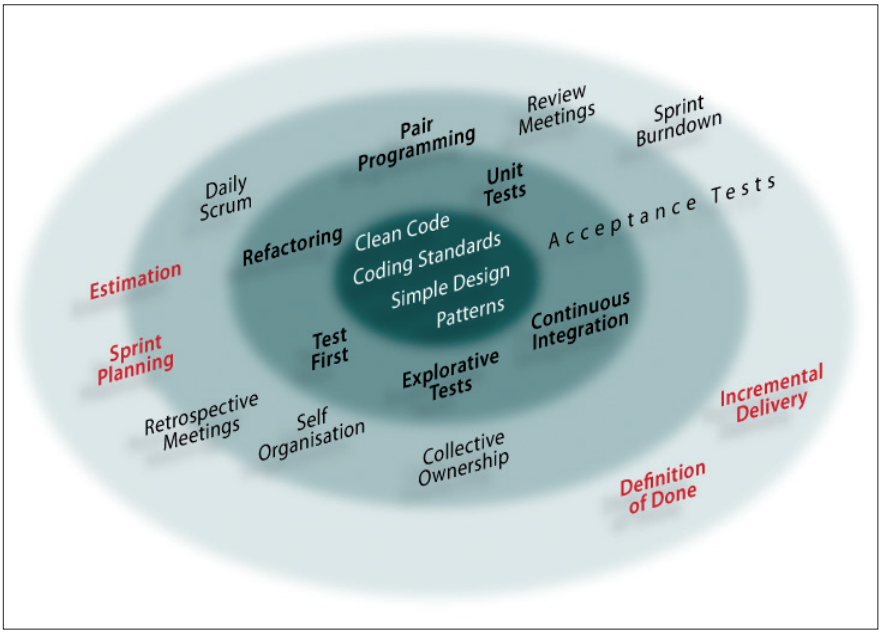
Das impliziert allerdings den Verzicht auf eine allzu große Betonung der Individualität innerhalb des Teams. Sehr deutlich wird das im ASE-immanenten Grundsatz des Pair Programming: Wann immer ein Entwickler eine Zeile Code eintippt, sitzt ein zweiter Entwickler neben ihm und reflektiert, was er sieht.

Hintergrund für dieses Vier-Augen-Prinzip ist die Tatsache, dass die Korrektur von Fehlern in der Software umso weniger kostet, je früher in der Produktionskette der Fehler entdeckt wird.

Trotzdem zögern viele Unternehmen zunächst, Pair Programming einzuführen. Sie befürchten, ihre Entwicklungskapazität zu halbieren. Die Realität zeigt jedoch ein anderes Bild: Trotz des doppelten Entwicklereinsatzes sinkt die Produktivität beim Pair Programming meist nur um etwa zehn Prozent – eine sicherlich vertretbare Investition für die dauerhaft steigende Codequalität.



[Abb. 1] Die vier Spären des Agile Software Engineering Trainings.



[Abb. 2] Best Practices sorgen für Softwarequalität.

Nach anfänglicher Zurückhaltung unterstützen viele Entwickler das Pair Programming, weil sie über das Wissen ihrer jeweiligen Programmierpartner in immer mehr Gebieten profunde Kenntnisse erhalten. Außerdem gibt das gemeinsame Programmieren mehr Sicherheit.

Dennoch – wer sich mit dem Gedanken nicht anfreunden kann, der wird nach den ASE-Prinzipien auch nicht dazu gezwungen. Zumal es ohnehin Routineaufgaben gibt, bei denen Pair Programming schlicht unnötig ist.

Ein zweiter elementarer Bestandteil von ASE ist die testgetriebene Entwicklung.

Testgetrieben: Qualitätsmanagement ab der ersten Zeile

Qualitätsmanagement ist im ASE-Programm kein nachgelagerter Prozess, sondern beginnt mit der Formulierung von Anforderungen und vor allem von Akzeptanzkriterien.

In erster Konsequenz heißt das, dass die Software automatisch testbar sein muss und es keine undefinierten Systemzustände gibt. Das entspricht einer geistigen Trennung von Schnittstelle und implementiertem Objekt. Im Sinne der freien Austauschbarkeit der Objekte gilt für sie der Grundsatz des Verbergens von Informationen (information hiding) beziehungsweise der Kapselung (encapsulation). ASE verlagert die Testaufwände in die Entwicklung und sichert damit die konstante Rückkoppelung zwischen Programmierung und Anforderung.

ASE C#: Das Angebot für .NET-Entwickler

ASE ist ein Referenzrahmen, der über unterschiedliche Technologiewelten hinweg verbindlich eingesetzt werden kann, etwa über Java RCP, Java Web, C++ und ABAP. Seit Neuestem ist ASE auch für Microsoft .NET (C#) verfügbar.

Dabei orientiert sich die erste Stufe des ASE-C#-Trainings, das Training im Labor, stark am Kursangebot für Java Web.

Ein großer struktureller Unterschied liegt jedoch darin, dass prinzipiell jeder Kunde aus dem Microsoft-Umfeld die verwendeten Technologien anders zusammenstellen kann.

Aufgrund der resultierenden sehr unterschiedlichen Kombinationen ist es kaum möglich, jeden individuellen Technologie-Mix darzustellen. Deshalb legt andrena den Fokus auf ein Standardset, mit dem ein Großteil der .NET-Entwickler arbeiten kann. Außerdem gibt andrena bestimmte Code-Fragmente vor, um den Teilnehmerinnen und Teilnehmern den Zugang zum ASE-Programm so leicht wie möglich zu machen. Für den Lernerfolg wäre es kontraproduktiv, zuerst diverse Manuals eines Frameworks studieren zu müssen.

Konkret zum Einsatz kommen bei ASE C# folgende Technologien und Werkzeuge:

- Microsoft Visual Studio 2010
- Windows Presentation Foundation (WPF)
- Managed Extensibility Framework (MEF)
- MS Entity Framework 4, MS SQL Server Express
- Coded UI Tests, MS Test, Fitnesse (http://fitnesse.org/)
- Team Foundation Server (TFS)

Entwickelt wird unter Microsoft Visual Studio 2010, das für die meisten Teilnehmer die bekannte Entwicklungsumgebung sein dürfte. Nun ist das Refactoring von Programmcode ein wesentlicher Bestandteil der agilen Softwareentwicklung.

Deshalb installiert andrena zusätzlich die Visual-Studio-Erweiterung ReSharper, die über einen größeren Funktionsumfang verfügt.

Ganz bewusst fiel die Entscheidung zugunsten der Technologievariante von Microsoft selbst. Dessen ungeachtet sind selbstverständlich Alternativen möglich, zum Beispiel die Verwendung von NUnit statt MS Test oder von NHibernate statt des MS Entity Frameworks.

Die Offenheit hinsichtlich der verwendeten Technologien – und deren Kombinationen – ist ein grundlegendes Kennzeichen des Programms ASE C#. Denn dessen Lernziel besteht nicht darin, sich in konkrete NET-Technologien einzuarbeiten. Vielmehr geht es darum, dauerhaft im Team ein neues Qualitätsbewusstsein zu wecken.

Diese klare Qualitätsorientierung beginnt mit der Aufgabe, sauberen Code (Clean Code) zu schreiben, und reicht bis zur Optimierung des kompletten Softwareentwicklungszyklus, von der Anforderung bis zum fertigen Software-Inkrement. Die dazu erforderlichen Kenntnisse soll das ASE-Training vermitteln.

Einüben im Projekt

Im Gegensatz zum einwöchigen ASE-Training haben die Entwicklerinnen und Entwickler im Projektalltag nur selten mit neuen Projekten, geschweige denn Greenfields zu tun oder mit dem aktuellen Microsoft-Technologie-Stack. Damit die erlernten Inhalte trotzdem in die Praxis übertragen werden können, begleitet der Trainer das Team während drei einwöchiger Sprints in dem realen Projekt. Diese Praxis hat sich bei den ASE-Trainings für die anderen Technologie-Stacks bestens bewährt und wurde daher übernommen.

Allererste Aufgabe des Trainers ist es, zunächst Hindernisse der Art: „Wir können Scrum nicht einsetzen, weil …“, „Wir können hier keine Tests schreiben, weil …“ zu analysieren und gemeinsam mit dem Team Lösungen zu erarbeiten.

Anders ausgedrückt soll der Trainer dabei helfen, die existierenden Rahmenbedingungen so zu modifizieren, dass agiles Programmieren überhaupt möglich wird. Dazu können die ASE-Trainer schon auf einen großen Erfahrungsschatz von Lösungsvorschlägen zurückgreifen, die aus der langen Erfahrung mit agilen Methoden abgeleitet wurden.

Jeder Trainer unterstützt die einzelnen Entwicklerinnen und Entwickler darin, die erlernten Techniken und Prinzipien in den Arbeitsalltag zu integrieren, also Estimation, Testing, Clean Code, DoD-ready zu praktisch selbstverständlichen Bestandteilen der eigenen Arbeitsweise werden zu lassen.

Schon gegen Ende des Programms sind oft erste Ergebnisse der neuen Arbeitsweise im Projekt sichtbar – zum Beispiel genauere Schätzungen, weniger Bugs.

Fazit

Bei SAP nahmen bereits 2011 rund 1400 Entwicklerinnen und Entwickler am ASE-Trainingsprogramm teil. Wie erste Umfragen zeigen, ist eine deutliche Mehrheit der Teilnehmerinnen und -Teilnehmer sehr angetan.

Seit 2012 steht ASE jetzt auch anderen Unternehmen offen, die ihre Scrum-Teams stärken wollen. **[tib]**

[1] Kent Beck. Test Driven Development. By Example. Addison-Wesley Longman, Amsterdam, November 2002.

[2] Ken Schwaber and Mike Beedle. Agile Software Development with Scrum. Pearson, April 2008.

[3] Robert C. Martin. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 1st edition, August 2008.

[4] Gojko Adzic. Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing. Neuri Limited, United Kingdom, 2009.