➢ Fachbericht vom 21. April 2015

Testen in der agilen Softwareentwicklung – wichtig und möglich!

Eine fehlerhafte Programmzeile, ein falscher Datentyp oder ein Vorzeichenfehler bei einer Berechnung - und schon werden teure Fehler riskiert. Ein Buchungsfehler im Juni 2014 kostete die Hessische Landesbank rund 900.000 Euro (Quelle: Die Welt vom 06.06.2014). Im Jahre 1999 führte eine nicht funktionierende elektronische Sperre bei der WestLB dazu, dass mehreren Kunden Beträge nach bargeldloser Zahlung doppelt in Rechnung gestellt wurden (Quelle: Spiegel online vom 28.07.1999). Einfache und durch Tests vermeidbare Fehler. Doch wie sehen Tests in einem Projekt aus, in dem agil Software entwickelt wird und das keinem in Phasen strukturierten Plan folgt?

Foto: Beckmann & Partner CONSULT

Herausforderung agile Softwareentwicklung im Zahlungsverkehr

Dieser Bericht handelt von den Herausforderungen und den Erfahrungen aus der agilen Softwareentwicklung im Zahlungsverkehr, denen ich mich als Projektmitarbeiter in einem komplexen Zahlungsverkehrsprojekt gegenübergestellt sah.

Diese Herausforderungen liegen nicht nur in der Entwicklung von gewollten oder geforderten Funktionalitäten, sondern auch in der Integration von vor- und nachgelagerten Geschäftsprozessen – und das zu jedem Zeitpunkt unter Sicherstellung der Funktionalität und Qualität. Um auf Kundenanforderungen, Auflagen der Bankenaufsicht, Änderungen an Schnittstellen oder Anpassungen an Prozessen flexibel reagieren zu können, kann die agile Softwareentwicklung eine Lösung sein.

Wird ein Projekt beispielsweise nach dem agilen Vorgehensmodell Scrum vorangetrieben, folgt die Entwicklung der Anwendung einem festen Rhythmus. In so genannten Sprints mit einer Dauer von wenigen Wochen wird eine vorher festgelegte Auswahl von Anforderungen umgesetzt. Am Ende des Sprints steht die Auslieferung

fertiger Funktionalitäten.

Softwaretests bei agiler Softwareentwicklung

Eine besondere Betrachtung bei agiler Softwareentwicklung verdient der Test. Wie groß schneidet man die Testobjekte? Welche Testdaten werden verwendet? Kann der Test unabhängig von der Entwicklung laufen? Ist es möglich, eine vollständige Testabdeckung zu erreichen?

Zeitversetztes Testen von Sprints

Eine Möglichkeit, den Softwaretest in die agile Entwicklung zu integrieren, ist die Zeitversetzung des Tests um einen Sprint: Während neue Funktionalitäten in einem Sprint entwickelt werden, werden die Funktionalitäten des letzten Sprints getestet. Der Fokus liegt so auf genau definierbaren Testobjekten, die eine schnelle Resonanz der Tester ermöglichen.
Um die Testabdeckung zu erhöhen, können diese Sprint-Tests durch eine Kombination von Entwicklertest, Komponenten-Test, explorativem Test und Performance-Test - auf die ich im Folgenden eingehe - während der Entwicklung unterstützt werden.

Auf der Suche nach Fehlerwirkungen - Entwicklertests

Den ersten Weg auf der Suche nach Fehlerwirkungen stellen Entwicklertests dar. Auf Grund seiner eigenen Erfahrung und seiner Kenntnis des Codes testet der Entwickler seine entwickelten Programmfunktionen. Entwicklertests können vereinzelt manuell durch den Entwickler durchgeführt werden. Oder, bei einem Build der Anwendung, auch im Ganzen automatisiert durchgeführt werden.

Strukturieren und Vereinfachen - mit Komponenten-Tests

Strukturiert und vereinfacht wird das Testen durch Komponenten-Tests: Einzelne Funktionalitäten werden gruppiert und gesammelt getestet - zum Beispiel der Import einer Datei. Werden diese Komponenten-Tests automatisiert zu bestimmten Ereignissen durchgeführt, zum Beispiel zu einem Build der Anwendung, erhält man eine einfache und wenig zeitaufwändige Qualitätssicherung.

Weitere Möglichkeiten – Explorativer Test, Performance-Test, Parallel-Produktion

Eine zusätzliche Testaktivität ist der explorative Test – ein erfahrungsbasierter Test, der keinem Testplan folgt, sondern Funktionalitäten auf Grund eines „Bauchgefühls"

des Testers kontrolliert. Außerdem der Performance-Test, der das Verhalten der Anwendung unter Last überwacht. Neben diesen Testaktivitäten zeigt die Einrichtung einer Parallel-Produktion überzeugende Ergebnisse: Die in der Produktions-Umgebung verarbeiteten Daten werden anonymisiert in die Test-Umgebung eingeleitet und verarbeitet.

Vereinfachung der Tests durch Automatisierung - Systemtests

Erheblich minimieren lässt sich der Testaufwand durch automatisierte Systemtests: Ein festgelegter Fundus von Systemtests, die möglichst viele Funktionalitäten abdecken, werden täglich, auf dem aktuellen vom Entwickler freigegebenen Entwicklungsstand, ausgeführt. Dies hat einige Vorteile: Fehler werden so früh wie möglich entdeckt, wodurch der Entwickler noch nah am Kontext ist und eine Behebung des Fehlers kaum Aufwand bedeutet. Weiterhin wird die Aufdeckung von Fehlern über den gesamten Sprint verteilt, was zu einer moderaten Zusatzbelastung der Entwickler führt.

Bei diesen automatisierten Systemtests sind zwei wichtige Punkte zu beachten: Erstens ist eine vorherige Dokumentation des Testfalls unabdingbar. Möglicherweise deckt ein Testfall längere Zeit keine Fehler auf. Um dann bei einem möglichen späteren Fehlschlag des Tests diesen nachvollziehen und Fehler identifizieren und beheben zu können, ist die Dokumentation Voraussetzung. Zweitens sollten Testfälle, die nicht primär die GUI einer Anwendung testen, auch möglichst unabhängig von dieser sein. Deshalb sollten schon bei der Entwicklung einer Anwendung Systemschnittstellen für Tests geschaffen werden. Denn sonst können Änderungen in Texten einer Web-Anwendung, zum Beispiel das Verschieben oder Umordnen von Buttons, einen Testfall zum Fehlschlag bringen, obwohl die fachliche Ausführung weiterhin korrekt war.

Die richtigen Testdaten machen den Unterschied

Bei allen beschriebenen Tests spielen die benutzten Testdaten eine wichtige Rolle. In der Regel werden manuell erstellte Testdaten verwendet. Diese kann man genau auf das Testobjekt zuschneiden und so klein wie möglich halten. In einigen Fällen ist es aber unabdingbar, selbst entwickelte oder proprietäre Tools zur Testdaten-Generierung zu nutzen. So erfordert beispielsweise jedes Projekt im Bereich des Zahlungsverkehrs mindestens einen Lasttest, bei dem die Anwendung mit dem zukünftig erwarteten Volumen konfrontiert wird. Hierbei ist eine manuelle Erstellung von Testdaten ausgeschlossen. Eine Alternative kann in Einzelfällen der Test mit anonymisierten Produktiv-Daten darstellen: Die in einer Produktions-Umgebung verarbeiteten Daten werden zusätzlich in die Testumgebung eingeleitet. So wird nicht nur das produktive

Volumen mit wenig Aufwand simuliert, es wird zusätzlich auch eine breite Variabilität der Testdaten erzielt.

Um die oben beschriebenen Testaktivitäten durchzuführen, sind einige Voraussetzungen zu erfüllen: Neben der Einrichtung einer angemessenen Testumgebung mit der Möglichkeit der automatisierten Testdurchführung ist eine enge Zusammenarbeit zwischen Testern und Entwicklern notwendig. Vor allem bei der Implementierung der automatisierten Tests müssen die Tester von den Entwicklern unterstützt werden. Deshalb ist es sinnvoll, wenn die Tester Bestandteil des Projekt-Teams sind und an allen Projektaktivitäten - zum Beispiel bei Scrum: Daily Scrum, Sprintplanung - teilnehmen. So erfahren sie frühzeitig von neuen Funktionen in der Anwendung und können sofort mit der Entwicklung von Testfällen beginnen. Zudem können sie erwartete Aufwände kommunizieren, die durch Testaktivitäten entstehen.

Ansprechpartner:
Daniel Schlüter
Informatiker
ISTQB Certified Tester – Foundation Level

Beckmann & Partner CONSULT
Telefon: 0521 2997320
dsr@beckmann-partner.de
www.beckmann-partner.de

Quellen & Verweise:

Späte Überweisung kostet Landesbank eine Million, Die Welt vom 06.06.2014,
http://www.welt.de/wirtschaft/article128803010/Spaete-Ueberweisung-kostet-Landesbank-eine-Million.html

Buchungspanne: "Chaos auf dem Konto", Spiegel online vom 28.07.1999,
http://www.spiegel.de/wirtschaft/buchungspanne-chaos-auf-dem-konto-a-33183.html

ISTQB®/GTB Standardglossar der Testbegriffe Deutsch/Englisch
http://www.german-testing-board.info/service/information/glossar.html