

TESTEN VON REGRESSIONSTESTS – EINE FALLSTUDIE

B. Burger, UC4 GmbH, 3012 Wolfsgraben. Österreich

ABSTRACT

In diesem Beitrag beschreiben wir eine Methode zur Qualitätskontrolle von Regressionstests und ihre beispielhafte Anwendung bei der UC4 Software GmbH. Hierbei wurden spezifische Fehler in das zu testende Produkt eingebaut und die Auffindung bzw. Nicht-Auffindung der einzelnen Fehler durch das Testteam in weiterer Folge analysiert. Die Beitrag beschreibt die eigentliche Methode der Einbringung der Fehler sowie die Analyseschritte und Folgerungen, welche in der Folge gesetzt und gezogen wurden. Weiters geben wir entsprechende Hinweise zur Anwendung dieser Methode im Allgemeinen sowie zur menschlichen Dimension des Prozesses, welcher besondere Beachtung finden sollte.

EINLEITUNG

Es ist eine allgemein bekannte Tatsache, dass Regressionstestfälle mit zunehmendem Alter ihre Effektivität verlieren. Die Ursachen hierfür liegen unter anderem an der Weiterentwicklung des Produktes, sodass diesem weitere Funktionalität hinzugefügt wurde oder aber sich die Implementation bestehender Funktionalität änderte. Auch die Unterstützung zusätzlicher Plattformen mit ihren jeweiligen Eigenheiten kann dafür sorgen, dass das durch die Regressionstestfälle geknüpfte Sicherheitsnetz löchriger wird.

In diesem Beitrag stellen wir nun eine Methode vor, wie sie in einem ersten Experiment bei der UC4 Software GmbH zur Qualitätssicherung von Regressionstestfällen angewendet und in der Folge in die üblichen QA Maßnahmen integriert wurde. Dieses Experiment wurde durch die folgenden Fragestellungen motiviert.

1. Können wir sicher sein, dass unsere Regressionstests alle Schlüsselbereiche unseres Produkts abdecken, nachdem seit der Erstentwicklung mehr als 7 Jahre und unzählige Produktversionen ins Land gezogen sind?

2. Haben Prozessschwächen oder gar fehlende Prozesse zu Fehlern oder Lücken in den Regressionstestfällen geführt, sodass möglicherweise nicht alle Schlüsselbereiche des Produkts durch Testfälle abgedeckt sind?
3. Sind unsere Prozesse zur Testfallpflege gut genug, sodass Testfälle, welche im Rahmen der Produktentwicklung oder aber als Antwort auf Kundenprobleme entstanden sind, auch entsprechend in das Testfallportfolio eingearbeitet werden?
4. Sind wir sicher, dass wir die Kernfunktionalitäten noch richtig testen, obwohl sich deren Implementation innerhalb der letzten Jahre geändert hat?
5. Ist die von uns verwendete „Osterei“ Methode dafür geeignet, diese Fragen zu beantworten?

Um diese Fragen zu beantworten führten wir ein Experiment durch, dessen Aufbau und Methodik in den nächsten Abschnitten erläutert wird.

METHODE

Um die „Osterei“ Fehler im Produkt zu platzieren, wurde diese durch den Entwicklungsleiter in den Code eingebracht. Insbesondere wurde darauf geachtet, dass die eingebrachten Fehler nur auf den Maschinen des Testteams auftreten konnten. Daher wurden diese Fehler in Abstimmung mit dem QA Manager auf die zum Test vorgesehenen Maschinen eingeschränkt. Als weiteren Sicherheitsmechanismus wurde ein Zeitfenster angegeben, welches im Bereich des geplanten Regressionstests lag. Eine Zeitabfrage in der Fehleroutine verhinderte, dass die eingebrachten Fehler außerhalb dieses Zeitfensters ausgelöst wurden. Es wurden des weiteren entsprechende Kommentare in den Sourcen angebracht, welche die Entwickler instruierten, keine Veränderungen an diesen Code Segmenten vorzunehmen, ohne vorher mit dem Entwicklungsmanagement Rücksprache zu halten.

Um die Entwicklungsmanager in ihrer Entscheidung zu unterstützen, welche Fehler in das Produkt einzubringen wären, entwickelten wir eine Liste mit Selektionskriterien, wie sie in Tabelle 1 dargestellt ist. Wir halten diese Tabelle für allgemein gültig, wenn auch nicht für vollständig.

Selektionskriterien	Fehlerziel
Funktionalitäten, bei der Fehler leicht übersehen werden.	Testfall Design, Fehler in der Wahrnehmung der Tester.
Funktionalitäten, welche über mehrere Pfade erreicht werden können (bspw. via Icon, Hotkey und Kontextmenü)	Testfall Design. Vollständigkeit der Pfade
Funktionalitäten, welche in der Vergangenheit des öfteren erweitert bzw. verändert wurden.	Ungetestete Funktionen innerhalb eigentlich getesteter Funktionalität, Prozess der Testfallwartung
Funktionalitäten, welche aufgrund von Kundenproblemen verändert bzw. korrigiert wurden	Prozessschwächen in der Testfallwartung.
Funktionalitäten, welche mindestens zwei Versionen vor dem "Einpflanzen" der Fehler in das Produkt kamen	Prozessschwächen in der Testfallwartung

Tabelle 1: Selektionskriterien zur Auswahl einzubringender Fehler anhand gesuchter Schwächen und Fehlerziele.

In ihrem Prinzip entspricht die verwendete Methode dem „defect seeding“, wie es in verschiedensten Standardwerken (z.B. [THA02]) beschrieben wird. Jedoch verwenden wir die Methode mit einem anderen Ziel, als dies üblicherweise gemacht wird, da wir nicht Testende Kriterien festlegen, sondern Prozess- und Testfallschwächen damit aufdecken wollten. Die Einbringung der Fehler entspricht dem Prinzip von Mutationstests [LIG02]

Einen wichtigen Faktor bei dieser Methode stellt die „Blindheit“ der Tester dar, da nur so sichergestellt werden kann, dass der Regressionstest in typischer Weise ausgeführt wird und die Tester keinen „speziellen Testaufwand“ treiben. Daher wurden weder Tester noch Entwickler von diesem Experiment informiert. Als QA Manager und Initiator wusste ich zwar von der Einbringung der Fehler,

jedoch war mir nicht bekannt, wo und wie viele Fehler nun eingebracht wurden.

Die Regressionstests wurden innerhalb der Regressionstestphase in typischer Weise durchgeführt. Am Ende der Testphase wurden die Tester von dem Experiment informiert. Im Rahmen einer Nachbesprechung wurde jeder der eingebrachten Fehler vorgestellt und in Hinsicht auf die folgenden Fragen analysiert.

1. Wurde der Fehler gefunden?
2. Wenn ja, geschah dies durch einen direkt darauf ausgerichteten Testfall?
3. Wenn nein, gibt es einen Testfall, der diesen Fehler hätte finden können?
4. Wenn die Antwort zu 3 ja war: woran scheiterte die Detektion?
5. Wenn die Antwort zu 3 nein war: weshalb existiert hierzu kein Testfall?

Die Antworten zu diesen Fragen verwiesen und der folge direkt auf vergangene oder bestehende Schwächen in Testfalldesign und Prozessen.

EXPERIMENTELLER AUFBAU – DAS UNTERSUCHTE SYSTEM UND DIE EINGEBRACHTEN FEHLER

Um die Art der Fehler zu verstehen, welche im Rahmen des Experiments in das Produkt eingebracht wurden, erscheint es notwendig, einen kurzen Überblick über das getestete System zu geben.

Bei UC4:global handelt es sich um ein System für unternehmensweites, plattformübergreifendes Job Scheduling. Es ermöglicht dem Benutzer Jobs auf verschiedensten Plattformen zu planen, auszuführen und zu überwachen. Ebenso ermöglicht es die Steuerung und Überwachung von ERP Systemen wie SAP oder OA. Architektonisch besteht es aus drei Hauptkomponenten.

1. SERVER: Der Server ist das Herz der Applikation. In ihm erfolgt die gesamte Prozesssteuerung. Serverprozesse treten in zwei Ausprägungen auf, wobei eine die Aufgaben der Kommunikation zu den anderen Komponenten erledigt, während die andere für die Ablaufsteuerung zuständig ist. Der Server selbst läuft auf 6 verschiedenen Plattformen und unterstützt die Verwendung von drei verschiedenen Datenbanken.

2. EXEKUTOREN: Diese Programme laufen auf den einzelnen Maschinen, auf denen Jobs gesteuert werden sollen. Sie kommunizieren mit dem Server, starten Jobs, überwachen diese und berichten über Jobstatus und Ergebnis. Im Spezialfall eines Dateitransfers kommunizieren Exekutoren auch untereinander. Hierbei werden mehr als 20 unterschiedliche Plattformen unterstützt.
3. DIALOG: In Java entwickeltes Front End zur Entwicklung, Verwaltung und Monitoring der Jobs. Kommuniziert mit den Kommunikationsprozessen des Servers.

Wie aus dieser Aufstellung leicht ersichtlich ist, stellt der Server die kritischste Komponente dar. Daher wurden die Fehler in diese Komponente eingebracht. Die Art der Fehler und ihre jeweilige Motivation sind in Tabelle 2 beschrieben.

Fehler Nummer	Fehlerbeschreibung	Motivation
1	Passwortprüfung liefert "Wahr" unabhängig von der Eingabe.	Aus Beobachtungen heraus schien diese Routine ungetestet, da die Tester üblicherweise keine Passwörter vergeben, und auch der Testrobot kein Passwort verwendet.
2	Jobs werden 1 Stunde zu früh gestartet.	Zeitzonebetrachtungen stellen eine Kernfunktion des Produkts dar.
3	Unix Jobs liefern den Ende Status "Ended_OK", unabhängig vom eigentlichen Ergebnis-	Nachdem das Produkt eine große Anzahl von Plattformen unterstützt, sollte hierbei vor allem die Vollständigkeit der Testroutinen überprüft werden.

4	Die Berechtigungsprüfung für Schreibberechtigungen liefert „Wahr“, unabhängig von den eigentlichen Berechtigungen	Das Berechtigungssystem im Produkt ist äußerst komplex gestaltet und daher fehleranfällig. Insbesondere wurden kurz vor dem Test einige Probleme durch Kunden gemeldet.
5	Das Maximum parallel laufender Jobs wird nicht geprüft, wenn sie nicht innerhalb eine Kontrollobjekts (Jobplan, Schedule) gestartet werden.	Da verschiedenste Methoden zur Laufzeitkontrolle und zum Load balancing im system vorhanden sind, sollte hier die Vollständigkeit der Tests überprüft werden.
6	Die Synchronisationsroutine prüft nur die erste der Bedingungen.	Dieser Fehler trat ein paar Monate vorher bei einem Kunden auf. Hier sollte vor allem der Prozess der Testfallwartung getestet werden

Tabelle 2 Liste der eingebrachten Fehler und Motivation zur Einbringung

AUSWERTUNG UND ERGEBNISSE

Um die Ergebnisse detailliert und objektiv bewerten zu können, definierten wir eine Bewertungsskala, welche in Tabelle 3 dargestellt ist. Grundlage dieser Skala sind zwei Prämissen.

1. Jede Art der Fehlerdetektion ist generell ein positives Ergebnis.
2. Eine Prozessschwäche wirkt sich deutlich mehr negativ auf die Produktqualität aus als das Nichtfinden eines Fehlers aufgrund von fehlerhaftem Testfalldesign.

Unter Verwendung dieser Skala gelangten wir zu Tabelle 4, in der die Auffindung oder Nichtauffindung der einzelnen Fehler, die Ursache für die (Nicht-)Detektion und die entsprechende Bewertung aufgeführt ist.

Bewertung	Ergebnis
++	Der Fehler wurde durch einen Testfall gefunden, der direkt darauf abzielte.
+	Der Fehler wurde durch einen Seiteneffekt in einem Testfall gefunden.
-	Die Ursache des Nichtfindens lag im Testfalldesign
--	Der Fehler wurde aufgrund einer Prozessschwäche nicht entdeckt.

Tabelle 3 Bewertungsskala der Fehlerdetektion

Wie in Tabelle 4 beschrieben, wurden nur drei der 6 Fehler durch die Regressionstests gefunden. Obwohl auf den ersten Blick ernüchternd, darf auf der positiven Seite vermerkt werden, dass die gefundenen Fehler sehr früh innerhalb des Regressionstests gefunden wurden, bei dem ein Zyklus rund 5 Tage in Anspruch nimmt. Betrachten wir nun die einzelnen Fehler im Detail.

Fehler 1 wurde durch einen Seiteneffekt gefunden. Hierbei wurde durch den Testroboter der gesamte Ablauf der Benutzeranlage getestet. Aufgrund der nun fehlerhaften Prüfroutine wurde ein vom Testfallskript erwarteter Dialog nicht aufgerufen. Dieser Fehler brachte die Tester dazu, dies näher zu untersuchen, wodurch der Fehler in der Folge entdeckt wurde. Die Fehler 2 und 3 wurden durch direkt auf sie abzielende Testfälle gefunden.

Fehler Nummer 4 wurde nicht gefunden, obwohl Testfälle zur Prüfung dieser Funktionalität vorhanden waren und ausgeführt wurden. Es stellte sich heraus, dass bei diesen Testfällen Designfehler gemacht wurden, welche die Detektion verhinderten. Weitere Diskussion und Untersuchung führte auch auf entsprechende Prozessschwächen, welche diese Designfehler verursachten. Fehler 5 und 6 wurden beide aufgrund von Prozessschwächen nicht gefunden. Die Art der Prozessschwächen und deren Behebung beschreiben wir im nächsten Abschnitt.

Fehler	Bewertung	Dauer	Kurzbeschreibung des Ergebnis
1	+	1 d	Fehler wurde durch Testrobot gefunden, da ein erwarteter Dialog nicht aufgerufen wurde.
2	++	2 d	Ein Test Schedule mit vorgegebenen Ablaufzeiten wurde falsch exekutiert.
3	++	2 d	Die Prüfung von Testjobs zeigte falsche Ergebnisse und führte zur Fehlerdetektion.
4	- / --	n/a	Die vorhandenen Testfälle deckten das entsprechende Szenario nicht ab.
5	--	n/a	Die Testfälle für diese Funktionalität wurden nach ihrer Erstimplementierung nicht in das Regressionsportfolio übernommen.
6	--	n/a	Bestehende Testobjekte wurden nicht ausgeführt, da die Testfalldokumentation nicht entsprechend angepasst worden war.

Tabelle 4 Bewertung der Fehlerfindung mit Kurzbeschreibung der Urasche der (Nicht-)Detektion. Die Spalte Dauer gibt an, zu welchem Zeitpunkt im Testzyklus der Fehler gefunden wurde

DISKUSSION

Wie schon im vorhergegangenen Abschnitt erwähnt, konnte das Nichtauffinden von Fehlern in allen Fällen auf bestehende oder früher vorhandene Prozessschwächen zurückgeführt werden. Im Falle des Fehlers Nummer 4 lag die Nichtdetektion daran, dass die Testfälle zu dieser Funktionalität erst Jahre nach der Implementation entstanden. Diese Situation wurde noch durch die Tatsache verschlimmert, dass dem Testteam keine Anforderungsdokumente oder technischen Beschreibungen dieser Funktionalität zur Verfügung standen.

Daher orientierten sich die Tester beim Testfalldesign an der Produktdokumentation. Diese jedoch beschreibt nicht den technischen workflow dieser Funktionalität, welche in großen einer Anzahl von Szenarios bestimmte Code Teile überspringt. Für die Tester war es daher nicht feststellbar, dass die von Ihnen entworfenen Szenarien entweder den entsprechenden Code nicht durchliefen oder aber nicht zur Fehlersituation führten. Das fehlerhafte Testfalldesign konnte also direkt mit einer früheren Prozessschwäche, nämlich einer mangelhaften Dokumentation, in Verbindung gebracht werden. Neben dem inzwischen eingeführten Entwicklungsprozess, welcher die verpflichtende Erstellung eines Anforderungsdokuments und eine technischen Designdokuments erfordert, konnten wir auch eine Prozessverbesserung formulieren, welche eine Lösung für „dokumentationsarme“ Testfallerstellung darstellt.

Sollte entsprechende Dokumentation fehler, so hat ein „Grey Box Test“ Verfahren unter Mithilfe der Entwicklung angewendet zu werden.

Die Fehler 5 und 6 wiesen ebenso auf Prozessschwächen hin. Die Ursache für deren Nicht-Detektion liegt hierbei in der Entwicklung der Firma. Hierbei wurde in früherer Zeit der Test nicht als kontinuierlicher Prozess verstanden, sondern nur als Phase in einem Entwicklungsprojekt. Daher wurden zwei wesentliche Prozesse der Testfallwartung nicht formalisiert, sondern fanden nur auf einer ad-hoc Basis durch einzelne Tester statt.

Der erste Prozess, welcher nun institutionalisiert wurde betrifft die Wartung der Regressionstestfälle nach einer Änderung bzw. Erweiterung bestehender Funktionalität. Hierbei wurde festgestellt, dass vor allem Testfälle zu neuen Funktionalitäten nach ihrer initialen Testung nicht die Aufnahme in das Regressionstestportfolio schafften. Hierzu wurde folgende Prozessverbesserung formalisiert.

Nach einer Versionsfreigabe wird ein Wartungszyklus für die Regressionstests durchgeführt. Hierbei werden entsprechend den Änderungen und Neuentwicklungen dieser Version die Regressionstestfälle angepasst bzw. ergänzt.

Der zweite Prozess, welcher institutionalisiert werden musste, betrifft die Änderung bzw. Erweiterung von Regressionstestfällen nach kritischen Kundenproblemen. In diesem Fall wurde diese Schwäche durch eine Reorganisationsmaßnahme verursacht. Diese besagte, dass ab 2005 die Verantwortlichkeit für das Testen von Hotfixes von den Testern auf die Entwickler überging. Obwohl diese Entscheidung in ihrer Essenz korrekt war, führte Sie ungewollt zu einem Prozessbruch, da hierdurch das Testteam von Information über kritische Fehler im System, die auf keinen Fall ein weiteres Mal auftreten sollten, abgeschnitten wurde. Daher wurde folgende Prozessverbesserung implementiert:

Nach der Erstellung eines Hotfix wird das Testteam über Ursache und Fehlerbehebung informiert. Das Testteam untersucht dieses Problem und passt die Regressionstestfälle entsprechend an.

Neben den offensichtlichen Lehren, die aus dieser Untersuchung gezogen werden können, ziehen wir auch noch folgenden Schluss aus diesem Experiment. Die hier vorgestellte Methode ist geeignet für die Prüfung von Regressionstestfällen und ist ebenso dafür geeignet, Prozessschwächen in diesem Bereich aufzuzeigen. Sie ist des Weiteren kosteneffizient, da der Aufwand für das Einbringen der Fehler, die zusätzliche Untersuchungs- und Testzeit sowie Nachbesprechungen bei nur rund 15 Personentagen lag.

ABSCHLUSSBEMERKUNG: DER FAKTOR MENSCH

Zusätzlich zu den prozesstechnischen Punkten möchten wir auch auf entsprechende Fragen ansprechen, welche den psychologischen Effekt dieser Methode auf ihre QM Organisation betrifft. Unsere Ansicht nach ist es von äußerster Wichtigkeit bei den Nachbesprechungen herauszustellen, dass es sich bei der hier angewandten Methode nicht um ein Mittel zur Kontrolle der Tester, sondern um ein Mittel zur Kontrolle der Prozesse handelt. Können an diesem Punkt die Mitarbeiter des QM davon nicht überzeugt werden, kann diese Methode kein weiteres Mal eingesetzt werden, da die Tester dann zu anderen Mitteln greifen werden, die eingebrachte Fehler zu finden, anstatt die Regressionstestfälle anzuwenden.

Unter Betonung dieses Punktes erhielten wir durch unsere Tester ein positives Feedback, da sie diese Methode als ein Werkzeug sehen, mit dem sie ihre Arbeit verbessern können. Nichtsdestotrotz zeigten sie jedoch auch Skepsis in Hinblick auf „unnötige Mehrarbeit“ durch die eingebrachten Fehler. Auch aus diesem Grund empfehlen wir, nicht mehr als 2 bis 3 Fehler pro Tester in das Produkt einzubringen.

DANKSAGUNG

Der Autor erlaubt sich bei Hrn. Andreas Hammerschmied von UC4 Software für die Durchführung des Error Seedings und wertvolle Diskussionen zu bedanken. Ebenso sei Hrn. Siegfried Zopf für die wertvolle Diskussion gedankt.

REFERENZEN

THA02 G.E. Thaller, „Software-Test, Verifikation und Validation“, Heise, 2002

LIG02 P. Liggesmeyer, „Softwarequalität“, Spektrum Akademischer Verlag, 2002