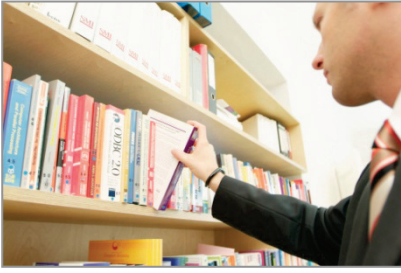


ANECON Test Service

Der Migrationstest-Prozess



KURZBESCHREIBUNG

Ein Softwaremigrations-Projekt erfordert einen anderen Testansatz als ein Entwicklungsprojekt. Denn es existiert im Regelfall weder eine Anforderungsspezifikation noch ein Designmodell als Testgrundlage. Das bedeutet, dass für Migrationsprojekte anforderungs- und modellbasierte Testansätze nicht geeignet sind. Vielmehr kann für den Test migrierter Systeme das Altsystem als Testorakel herangezogen werden. Das entscheidende Testkriterium ist dann die vollständige funktionale Gleichheit von Alt- und migriertem System - und dafür müssen spezielle Methoden und Werkzeuge zum Einsatz kommen.

ARTIKEL

Es gibt Projektarten, bei denen Testaufwand und Zeitaufwand berechenbar sind und daher dem Kunden für die Testabwicklung ein Fixpreis angeboten werden kann. Zum Beispiel dann, wenn das erzeugte Produkt gegen die Anforderungsspezifikation getestet werden kann. In diesem Fall handelt es sich um ein sauber spezifiziertes, stabiles Entwicklungsprojekt. Um zu einer Aufwands- und Kostenschätzung zu gelangen, können diese Anforderungsdokumente manuell analysiert werden. Alternativ besteht aber auch die Möglichkeit, diese Spezifikationsanalyse automatisch mittels eines Textanalyse-Werkzeuges durchzuführen, indem Daten (z.B. function- oder test points) extrahiert werden.

Ein anderer Projekttyp sind Wartungs- oder Reengineering-Projekte, bei denen die Software selbst nur leicht abgeändert wird. Für solche Projekte kann der Testaufwand auf Basis der Differenz zwischen Original- und geänderter Version geschätzt werden.

Ein davon abweichender Projekttyp sind hingegen die Migrationsprojekte, bei denen das ganze System gegen das Altsystem getestet wird. Die entscheidende Grundlage für die Aufwandsschätzung ist hier der Quellcode des Vorgängersystems [1].

Ein Migrationstest lässt sich am besten schrittweise aufbauen. Am Beginn muss die Messung jenes Quellcodes stehen, der migriert werden soll. Dies ist erforderlich, um exakt herauszufinden, was und mit welchem Aufwand getestet werden muss. Erst wenn die Software einer Messung unterzogen worden ist, ist es möglich, den nächsten Schritt im Migrationstest-Prozess auszuführen: den Test zu planen. Um diese Fehleraufdeckung bereits in der Anforderungsanalyse zu erreichen, bietet ANECON zwei Verfahren für das Prüfen der Artefakte:

Dieser zweite Schritt der Testplanung beinhaltet einerseits die Festlegung des Budget- und des Zeitrahmens; andererseits werden zu diesem Zeitpunkt auch bereits die einzelnen durchzuführenden Arbeitsschritte und die zu erwartenden Ergebnisse definiert. Mit dem manuellen Verfahren werden Dokumente, Entwurfsdiagramme und Codebausteine von QS-Experten mit Hilfe von Checklisten kontrolliert:

Der dritte Schritt ist das Design des Migrationstests. Es ist von großer Wichtigkeit, dass zu diesem Zeitpunkt jedes vom System verwendete Testobjekt identifiziert wird. Dazu zählen Benutzerschnittstellen, Datenbanktabellen, Systemschnittstellen und Berichte. Ebenso ist in diesem Prozessschritt das Testvorgehen – d.h. die einzelnen Testschritte und deren Reihenfolge – festzulegen.

Im vierten Schritt werden die Testdaten generiert. Üblicherweise werden die Testdaten für eine Migration aus dem Datenbestand der Produktion abgezogen und anonymisiert. Das dient dazu, den Datenschutzanforderungen zu genügen sowie um gegebenenfalls Privatsphären (z.B. im Falle von Kundendaten) zu schützen. Dieser Abzug von Produktionsdaten passiert, indem man das Altsystem in einer kontrollierten Umgebung laufen lässt. Dabei werden sowohl Eingabe- als auch Ausgabewerte sowie die jeweiligen, aktuellen Datenbank-Zustände aufgezeichnet und festgehalten. Die größte Herausforderung liegt darin, hohe Datenvolumina (wie sie eine Produktionsdatenbank oft mit sich bringt) korrekt zu handhaben. Genauso wichtig ist es, mit der Vielzahl an Importen, Exporten, Berichten und der Aufzeichnung der Benutzerinteraktionen sorgfältig umzugehen.

Mit den in Schritt 4 generierten Daten wird im fünften Schritt der eigentliche Migrationstest ausgeführt. In der Regel fehlen in einem Migrationsprojekt explizit definierte, ausformulierte Anforderungen. Detaillierte Testfälle müssen also über andere Wege abgeleitet werden.



Dazu bieten sich üblicherweise die Online-Transaktionen und Batchprozesse an. Die Menge an Dateninput, die in solche spezifischen Migrations-Testfälle fließen muss, kann eine beachtliche Größe annehmen und daher schwierig zu spezifizieren sein.

Das trifft gleichermaßen auf die Menge an Testfällen im Falle von Regressionstests zu. Konsequenterweise empfiehlt sich an dieser Stelle eine Automatisierung der Migrations-Testfälle - schließlich hat die Testdurchführung in einem Migrationsprojekt hauptsächlich mit der erfolgreichen Anwendung von Werkzeugen zu tun.

Im sechsten Schritt werden die Ergebnisse des Migrationstests evaluiert, um festzustellen, ob das migrierte System die Erwartungen erfüllt: nämlich dieselbe Leistung (inkl. Performance) wie das Altsystem zu erbringen. Dazu vergleicht man die Ergebnisse der beiden Systeme und hält jegliche Abweichungen als Nicht-Übereinstimmung fest.

Im siebenten und abschließenden Schritt wird entsprechend der im Testplan vereinbarten Ziele die Testabdeckung gemessen. So findet man heraus, ob der Programmcode durch die exekutierten Testfälle adäquat abgedeckt wurde.

Trifft dies zu, kann der Test abgeschlossen werden; andernfalls muss er solange fortgesetzt werden, bis die erwähnten Ziele erreicht worden sind. Diese Testziele (Abdeckung von Transaktionen, Code und Daten etc.) stellen die Basis der Testservice-Vereinbarung (TSA = test service agreement) dar – siehe Abbildung 1.

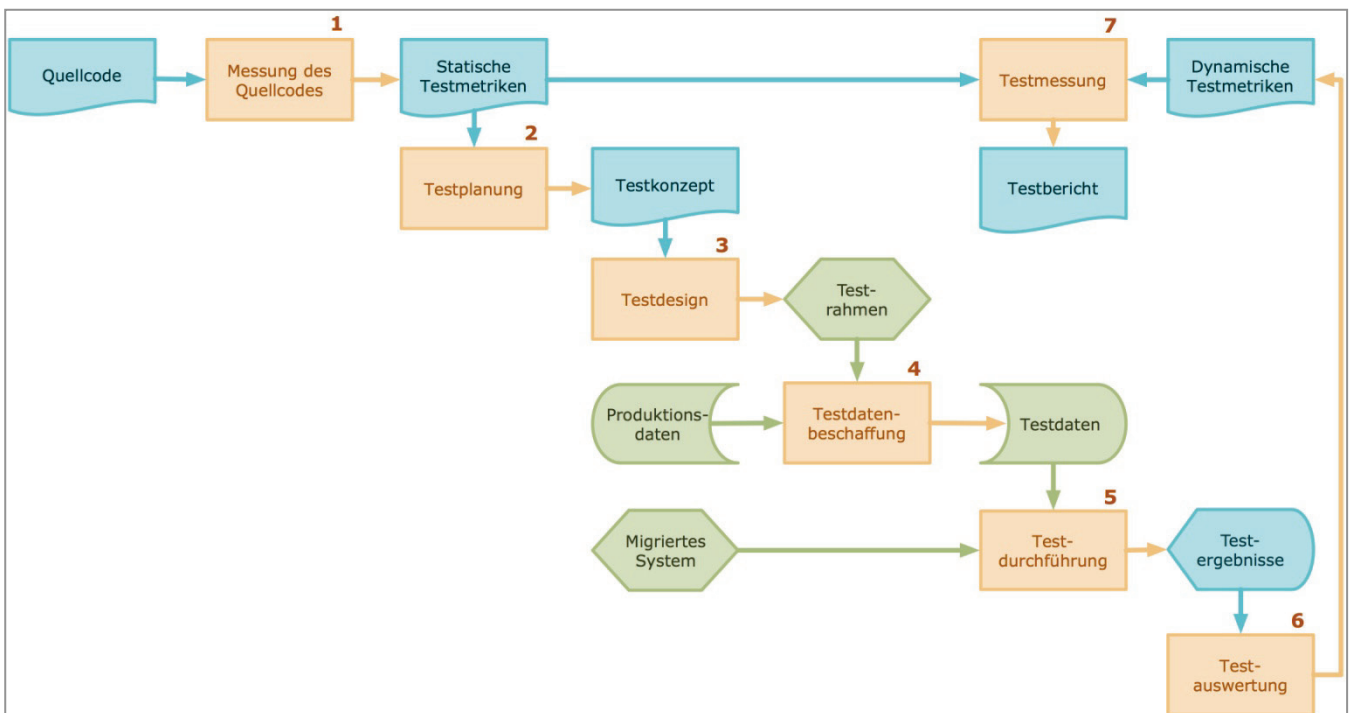


Abbildung 1



DIE MESSUNG DES CODES

„Man kann nicht planen, was man nicht messen kann“ [2]. Gemäß dieses Zitates von Tom deMarco muss im Rahmen der Planung eines Migrationstests damit begonnen werden, das zu messen, was migriert werden soll: den Code. Denn der Code ist es, der im Zuge einer Migration in eine andere Form transformiert wird. Soll zusätzlich auch noch ein Datenbestand migriert werden, müssen auch die betroffenen Datenstrukturen einer Messung unterzogen werden.

Der Code bietet verschiedene Bereiche, die gemessen werden können: die Komplexität des Kontrollflusses, der Datenverwendung, der Modulinteraktionen, der Wiederverwendbarkeit, der Wartbarkeit, der Konformität oder anderer Größen. Welche dieser Aspekte einer Messung unterzogen werden, hängt von der gewünschten Information und Zielgröße ab, die man mit der Messprozedur gewinnen will. Das bedeutet natürlich, dass wir uns hier z.B. mit der Anzahl der Code-Einheiten bzw. -Elementen, Datenbanktabellen und Datenschnittstellen auseinandersetzen. Denn diese müssen wir im Test berücksichtigen, um den funktionalen Anforderungen zu entsprechen. Die gewünschten Metriken können anhand statischer Analyse aus dem Quellcode, den Datenbank-Schemata, den Definitionen des Benutzerinterfaces sowie den Systemschnittstellen gewonnen werden. Die Messergebnisse zeigen dann an, wie viele Daten-Schnittstellen mit wie vielen Datenbank-Tabellen benötigt werden, um jede definierte Komponente testen zu können. Hier kann ein statisches Analysetool unterstützen, das die Testbarkeit des Systems auf einer rationalen Skala von 0 bis 1 errechnet [3].

Die Erfahrung vergangener Projekte zeigt, dass der Zeitaufwand für die Messung von Quellcode im Regelfall nicht mehr als 1 Woche beträgt; vorausgesetzt, man setzt dafür die geeigneten Messwerkzeuge ein. Einer der Autoren hat im Laufe der Zeit Werkzeuge entwickelt, die sich nicht nur für neue, sondern ebenso für alte Programmiersprachen eignen. Denn in den allermeisten Migra-

tionsprojekten ist das Vorgängersystem in einer alten Programmiersprache wie PL/1, Cobol oder C geschrieben.

DIE PLANUNG DES MIGRATIONSTESTS

Einen Migrationstest zu planen, umfasst unter anderem die Festlegung der Testziele, die Bereitstellung der Ressourcen sowie eine Zeit- und Aufwandsschätzung für die eigentliche Durchführung des Migrationstests. Der Testaufwand bemisst sich hier nach der Anzahl an zu exekutierenden Testfällen sowie jener an zu validierenden Testobjekten. Ein Testfall kann aus dem Nutzungsprofil der Produktionsumgebung abgeleitet werden und in Form einer bloßen Online-Transaktion, jedoch auch eines Ereignisses, eines Vorganges oder eines Batch-Prozesses definiert sein. Ein Testobjekt wiederum kann aus den Produktionsdaten generiert werden und ein Benutzer-Interface, eine Datei, eine Datenbank-Tabelle, eine Systemschnittstelle oder ein erstellter Report sein. Testfälle und Testobjekte zusammen stellen den Testumfang dar. Es ist erforderlich, diesen Umfang an der Testproduktivität aus vergangenen Projekten auszurichten. Auf diese Weise gelangt man zu einer Rohschätzung für das aktuelle Migrationstest-Projekt. Diese Schätzung kann mittels Produkttyp, der Produkt-Testbarkeits-Kennzahl, der Einflussfaktoren auf das Projekt und des Testwiederholungsfaktors verfeinert werden [4].

Das Ergebnis der Testplanung ist ein Testkonzept gemäß ANSI/IEEE-Standard 829 und eine Schätzung des Testaufwandes inklusive des zeitlichen Aspektes (Testdauer). Auf der Grundlage dieser 2 Parameter kann ein Preisangebot für den Migrationstest an den Auftraggeber gestellt werden. In diesem sollte besonderes Augenmerk auf die Testziele und die Testendkriterien gelegt werden.

Denn nur wenn diesen beiden Aspekten entsprochen wird, kann der Kunde den Test am Ende abnehmen. Umso wichtiger ist es daher, dass Testziele und Testendkriterien im Testkonzept in einfach messbarer Form angeführt werden. Das kann z.B. mittels Kennzahlen über die durchgeführten Testfälle und über die validierten Testobjekte erfolgen. Die Endversion des Testkonzepts kann ein standardisiertes Word-Dokument gemäß IEEE 829-Inhaltsverzeichnis sein [5].

DAS DESIGN DES MIGRATIONSTESTS

Um einen Migrationstest zu designen, muss man berücksichtigen, was und wie zu testen ist. Das „WAS“ lässt sich mittels Testfällen und Testobjekten (siehe oben) beschreiben. Diese können über eine Analyse des aktuellen Produktionsprozesses gewonnen werden. Dabei ist zu beachten, dass grundsätzlich bereits jede relevante Variante einer existierenden Transaktion einen potentiellen Testfall darstellt.

Leider kann man sich beim Design eines Migrationstests auf die Dokumentation des Altsystems nicht verlassen, da es selten auf dem letzten Stand oder oftmals gar nicht vorhanden ist. Die einzige Möglichkeit für einen Tester, den Test zu designen, ist es daher, sich mit den Endbenutzern des Systems zusammenzusetzen. Bei diesen Zusammenkünften zeichnet der Tester auf, wie die Benutzer aktuell das System bedienen. Dieses Vorgehen läuft auf eine nachträgliche Dokumentation der betrieblichen Nutzung hinaus. Ein solches schriftliches Belegmaterial stellt auch einen Großteil der Aufwände für die Regressionstests dar.

Die Testobjekte hingegen, die ebenfalls über Messaktivitäten aufgezeichnet werden, sind aufgrund ihrer Beobachtbarkeit leichter zu identifizieren. Von besonderer Bedeutung ist die Anzahl an Datenelementen, die jedes Datenobjekt besitzt.



Dazu zählen z.B. Felder, Spalten, Tags (sichtbare Zeichen zur Daten- und/oder Textstrukturierung) und Widgets (Komponente einer grafischen Benutzeroberfläche (GUI) bzw. ein kleines Programm, häufig nur zum Anzeigen einer Information). Ebenso relevant sind die Wertebereiche dieser Elemente, die man aus der Produktionsumgebung ablesen kann.

Die Testdatengewinnung erfolgt je nach Projektart aus unterschiedlichen Quellen. Diese Daten können den Testanforderungen entsprechend dann noch verändert und ergänzt werden.

Wie der Migrationstest entworfen wird, hängt einerseits davon ab, wie die Testobjekte tatsächlich identifiziert und validiert werden. Andererseits spielt hierfür auch eine Rolle, wie die Testfälle ausgeführt werden sollen.

Das Testdesign muss vorhersehbar machen, woher die Testobjekte gewonnen werden und mit welchen Mitteln die Testfälle aufgezeichnet werden können. Es sollte in einem strukturierten, semi-formalen Format vorliegen – dafür bietet sich z.B. der Einsatz einer Excel-Tabelle, eines grafischen Tools oder eines XML-Dokuments an (siehe Abbildung 2).

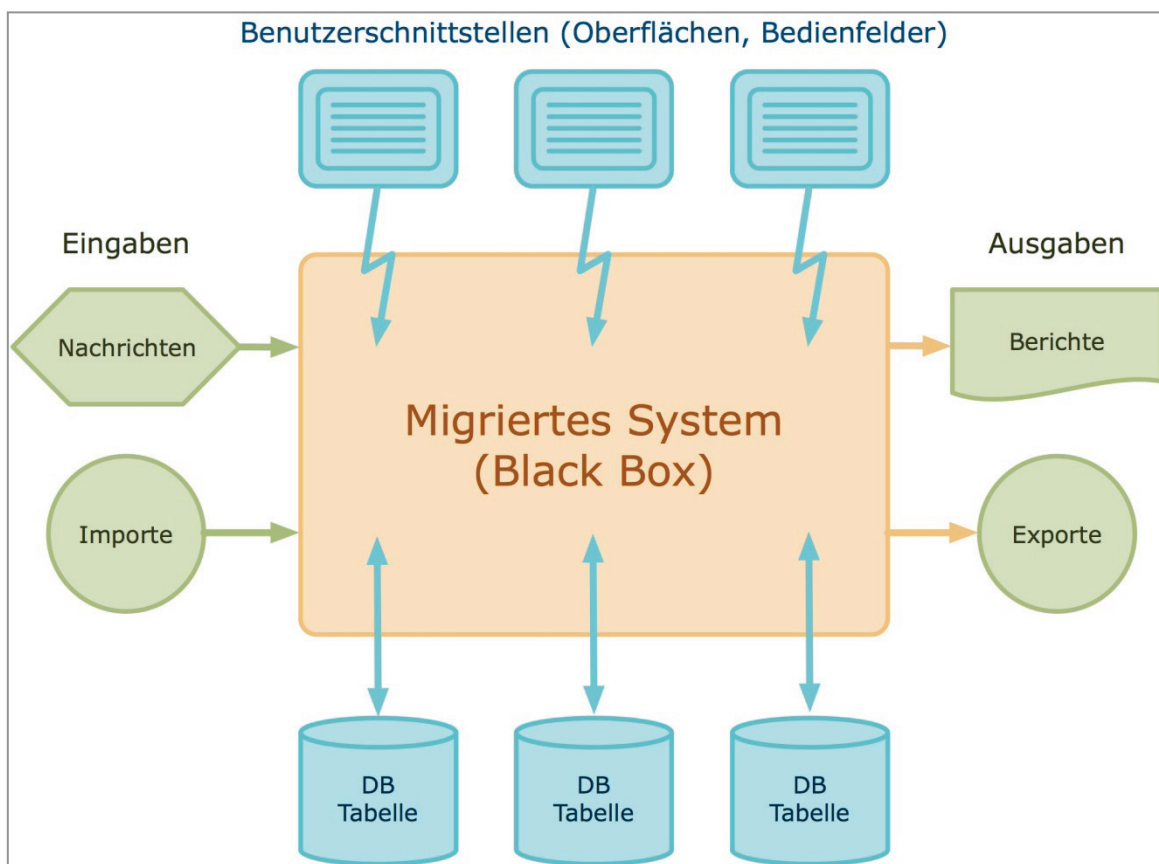


Abbildung 2

DIE GEWINNUNG VON TEST-DATEN FÜR EIN MIGRATIONSPROJEKT

Im Gegensatz zu den Tests in einem Entwicklungsprojekt muss man – wie bereits erwähnt – in einem Migrationsprojekt keine gesonderten Testdaten generieren, sondern gewinnt sie am besten direkt aus dem Produktionssystem. An dieser Stelle stellt sich dann jedoch die Frage: welche Daten ziehe ich aus diesem System dafür ab? Die gesamte Produktionsumgebung zu kopieren, ist im Regelfall nicht empfehlenswert und auch zu aufwändig. Im Testdesign hingegen hat man bereits einzelne Transaktionen, Ereignisse und Batchläufe mit repräsentativem Charakter identifiziert. Diese kann man nun unter Zuhilfenahme des existierenden Systems in einer kontrollierten Umgebung ausführen. Vor jedem Test müssen Abbildungen des Benutzer-Bedienfeldes, der Eingabedateien und des Datenbankinhaltes archiviert werden. In der gleichen Form müssen nach jedem Test die neuen Abbildungen derselben Objekte sowie zusätzlich gesendete Ausgabe-Nachrichten und ausgedruckte Reports festgehalten werden. All das führt zu einem großen Aufkommen an Testdaten, das sich in so manchen Vorgängerprojekten oftmals nur mit einem speziellen Server bewältigen und managen hat lassen.

Hier können einerseits Testwerkzeuge als nützliche Helfer zum Einsatz kommen:

- um Benutzer-Interfaces aufzuzeichnen
- um die vielen alten und neuen Daten-Abbilder managen zu können, die aufbewahrt werden müssen.

Andererseits bietet sich auch die Verwendung eines Capture-Replay-Tools an. War in der Vergangenheit keines verfügbar, haben sich die Autoren damit beholfen, die Benutzeroberfläche der Produktionsumgebung einfach über einen Screenshot festzuhalten, und diesen als Spezifikation für den Test des migrierten Systems zu verwenden.

Ob mit oder ohne Werkzeugunterstützung: das Problem der Datenbeschaffung für Migrationstests ist also primär jenes, umfangreiche Mengen an aufgezeichneten Daten zu verwalten. Jeder, der ein migriertes System – unabhängig von dessen Größe – testen möchte, muss dieser Aufgabe gewachsen sein [6].

DIE DURCHFÜHRUNG DES MIGRATIONSTESTS

Migrationsprojekte zeichnen sich speziell durch den Testaufwand in Relation zu anderen Aktivitäten aus. Wenn Systeme neu entwickelt werden, werden sie über einen langen Zeitraum Modul für Modul, Komponente für Komponente, System für System getestet. Jedes Modul muss entworfen, codiert und getestet werden. Da diese Prozesse stark miteinander verwoben sind, liegt nicht gleich auf der Hand, wieviel Gesamt-Testaufwand betrieben wird. Nur der Systemtest am Ende des Projektes hebt sich davon als reine Testaktivität klar ab.

In einem Migrationsprojekt ist das jedoch nicht der Fall, da hier zeitgleich ganze Systeme getestet werden müssen. Da die Code-Überführung meistens automatisiert von Personen durchgeführt wird, die mit der Funktionalität des Systems nicht vertraut sind, gibt es keine Möglichkeit, herauszufinden, wo Fehler auftreten werden. Wenn Code falsch transformiert wird, werden es diese Personen im Regelfall gar nicht bemerken. Damit ist es umso mehr die Aufgabe der Tester, die Korrektheit des migrierten Systems zu beweisen und Transformationsfehler aufzuspüren. Daher werden 60% bis 80% des Gesamtaufwandes in Migrationsprojekten dem Test gewidmet. Sofern der Migrationstest also nicht automatisiert durchgeführt wird, ist es unmöglich, ihn innerhalb eines vernünftigen Zeitrahmens zu absolvieren.

Damit man im Nachhinein feststellen kann, welcher Teil des migrierten Programmcodes getestet wurde, sollten Messfühler vor Teststart in jedem Zweig oder zumindest in jeder Methode oder jedem Programmcode-Block platziert werden. Für

diese Aufgabe können Automatisierungswerkzeuge genutzt werden. Bei Testbeginn wird dann das migrierte System schrittweise mit den Daten der alten Transaktionen bombardiert, wobei mittels Testtreiber der menschliche Benutzer simuliert wird. Wenn eine Transaktion fehlschlägt, wird nichtsdestotrotz die nächste gestartet. Nach jeder Transaktion oder jedem Batchprozess werden zudem die Inhalte der Ausgabefelder, der Ausgabe-Nachrichten und das neue Erscheinungsbild der betroffenen Datenbank für eine spätere Evaluierung aufgezeichnet. Es wäre nämlich zeitlich zu aufwändig, diese gleich während der Testphase zu validieren. Steht kein automatisiertes Aufzeichnungswerkzeug zur Verfügung, sollten die Inhalte der Ausgabe-Bildschirme zumindest manuell als Screenshot erfasst und für spätere Vergleiche abgespeichert werden. Denn das Hauptaugenmerk muss in dieser Phase auf einer möglichst unterbrechungsfreien Durchführung des Tests liegen. Treten Anomalien auf, werden sie ebenfalls festgehalten, um sie zu einem späteren Zeitpunkt zu überprüfen - denn für kreatives Testen ist in einem Migrationstest kein Platz [7].

DAS EVALUIEREN DES MIGRATIONSTESTS

Erst wenn die Durchführung des Migrationstests abgeschlossen ist, sollten seine Ergebnisse validiert werden. Zu Validierungszwecken werden die Ergebnisbilder des Migrationstests automatisch mit jenen aus den Tests des Altsystems verglichen. Nacheinander werden die Bildschirmhalte, Schnittstellendaten und Datenbank-Tabellen des migrierten Systems mit jenen des Vorgängersystems einem Vergleich unterzogen. Weichen die Formate voneinander ab, muss zugunsten einer Vergleichbarkeit eine Konvertierung in ein einheitliches Format stattfinden.

Zu diesem Zweck hat einer der Autoren mehrere Umwandlungs- und Vergleichs-Werkzeuge entwickelt. Die Benutzeroberflächen, die früher vielleicht Großrechner-Speicherabbilder oder UNIX-Bildschirme gewesen sind, werden in



XML-Dokumente transformiert. Die neuen Benutzeroberflächen (z.B. Webseiten oder Mashups (engl. „Verknüpfung“) – sie stehen für die Erstellung neuer Inhalte durch die nahtlose (Re-)Kombination bereits bestehender Inhalte) werden ebenfalls in ähnliche XML-Dokumente konvertiert. Auf diese Art und Weise können individuelle Datenelemente selektiv miteinander verglichen werden,

unabhängig von ihrer jeweiligen Position oder Darstellungsart am Bildschirm.

Gleichermaßen wird mit den Datenbanken verfahren. Die Inhalte der alten Datenbanken (IMS, IDMS, IDS, ADABAS, DB2, VSAM, sequentielle Dateien etc.) werden ebenso wie jene der neuen, relationalen Datenbanken heruntergeladen und dabei in komma-separierte (CSV-) Dateien umgewandelt.

Anschließend werden diese 2 CSV-Dateien Spalte für Spalte miteinander verglichen. Unterschiedliche Formate oder berechnete Werte können auch mittels definierter Regeln und Aussagen geprüft werden (siehe Abbildung 3).

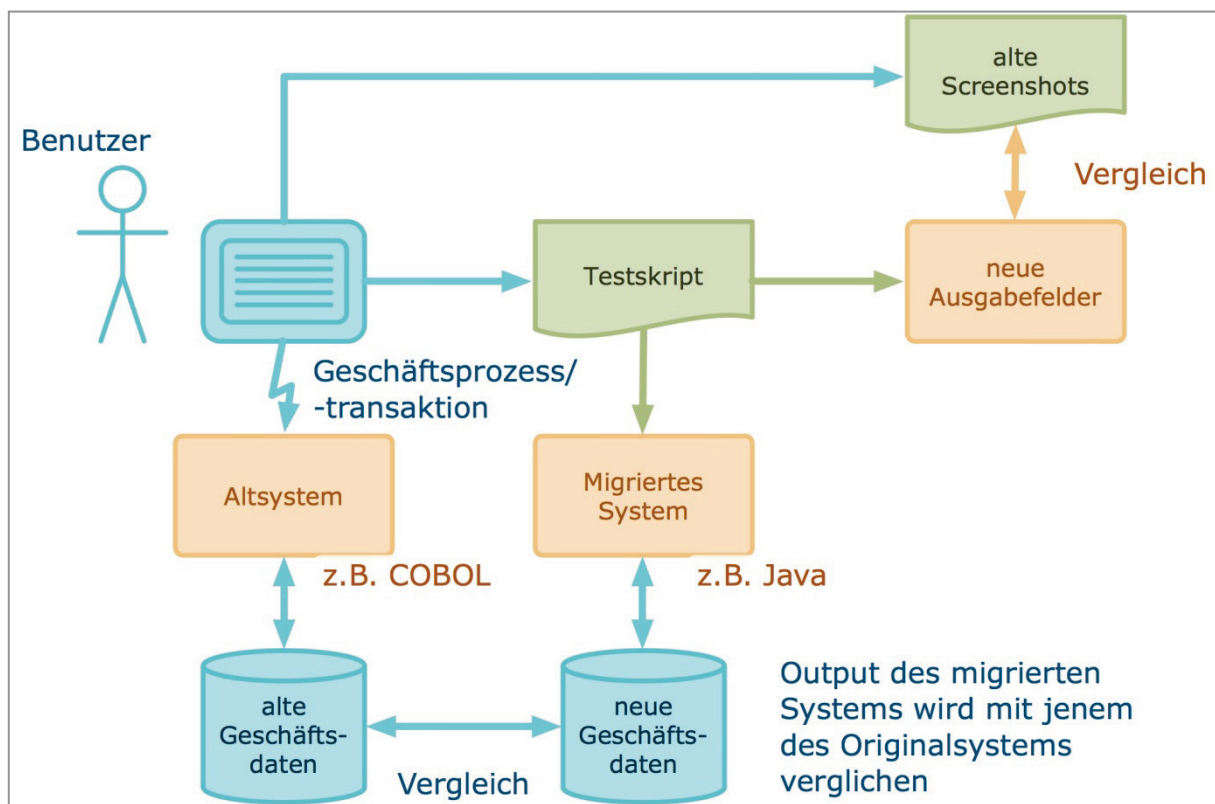


Abbildung 3

Im Zuge der Validierung von Reports und System-Schnittstellen kann ein Problem auftreten: dass diese in einem vollkommen anderen Format als die alten Versionen vorliegen können. Selbstverständlich sollten die Werte der einzelnen Datenelemente identisch sein: wenn der alte Rechnungsbetrag „EUR 19,99“ war, dann muss auch der neue Rechnungsbetrag exakt diesen Wert aufweisen. Nichtsdestotrotz kann dieser Wert z.B. am neuen Bildschirm oder Papiausdruck vollkommen anderswo platziert sein.

Um dieses Problem zu lösen, können Werkzeuge verwendet werden, die individuelle Datenelemente aus einem beliebigen Ausdruck, Bildschirm oder einer Textdatei extrahieren, und sie in eine XML-Datei schreiben. Die Namen und Typen von Datenelementen werden aus der ursprünglichen Datenbeschreibung in die Bildschirmsprache extrahiert. Als Vergleichsgrundlage dient ein XSD-Schema. Auf diese Weise können Daten von einer früheren Großrechner-Liste (im Sinne eines Programmausdrucks) nun mit Daten im XML- oder WSDL-Format verglichen werden.

Der Datenvergleich auf der Ebene dieser elementaren Einheiten enthüllt selbst minimale Unterschiede, z.B. wenn ein Dezimalkomma falsch positioniert ist. Im Falle des Vergleichs von Millionen von Datenausgaben ist der automatisierte Vergleich die beste Art und Weise, Fehler im migrierten Code aufzuspüren. Ein alternativer Weg ist es, die Ausführungspfade über den Code einander gegenüberzustellen. Diese Vorgehensweise erfordert jedoch ein Verfolgen der getesteten Transaktionen..

Aber auch dafür gibt es geeignete Werkzeuge wie z.B. TestDoc. Es zeichnet nicht nur die durchlaufenen Pfade des Programmiercodes auf, sondern sogar die Anzahl an getätigten Durchläufen der Methoden oder Prozeduren [8].

DIE MESSUNG IM MIGRATIONSTESTS

Das Monitoring der Testdurchführung ist eine wichtige Voraussetzung, um den Prozentsatz der Codeabdeckung korrekt messen zu können. Da es der Code ist, der konvertiert wird und nicht die Funktionalität an sich, ist es, der einer Messung unterzogen werden muss. Vor der Testausführung werden daher automatisch Messfühler oder Verfolgungspunkte im Code eingepflanzt. Sie werden dann während der Testausführung - sollten sie durchlaufen werden - in einer Log-Datei aufgezeichnet. Nach dem Testdurchlauf werden diese Log-Dateien ausgewertet, um herauszufinden, wie effizient der Test wirklich gewesen ist.

Reports können unterschiedliche Informationen dokumentieren: der eine enthält z.B. den Testabdeckungsgrad als Prozentzahl der durchlaufenen Messfühler, der mit dem ursprünglichen Testabdeckungsziel aus dem vereinbarten Dienstleistungsvertrag verglichen wird. Ein anderer Bericht wiederum dokumentiert den Pfad jeder einzelnen Testtransaktion, und zeigt damit an, welche Codeeinheiten in welcher Reihenfolge ausgeführt worden sind. Diese Ausführungspfade können dann auch mit jenen des Originalsystems verglichen werden, um gegebenenfalls zu bestimmen, wo etwas im neuen System schief gelaufen ist. In einem dritten Report werden dann

z.B. jene Codeeinheiten festgehalten, auf die bestimmte Transaktionen Auswirkungen haben. Sollte es erforderlich werden, für eine spezielle Codeeinheit (Methode oder Prozedur) eine Fehlerbehebung durchzuführen, trifft dieser Bericht eine Aussage darüber, welche Testfälle wiederholt werden müssen.

Beim Test eines migrierten Systems werden andere Fehlertypen als beim Test eines komplett neu entwickelten Systems auftreten. In letzterem werden möglicherweise Funktionen ausgespart, Datentypen falsch definiert, Geschäftsregeln falsch interpretiert, Ergebnisse falsch kalkuliert usw. Beim Test eines migrierten Systems sind die Fehler allerdings subtilerer Natur: Nachkommastellen gehen verloren, Datensätze werden zerstört, Daten haben sich an einen anderen Ort verlagert, Bedingungen haben sich umgekehrt usw. Die Ergebnisse mögen zwar richtig scheinen, sie sind es jedoch nicht. Die einzige Möglichkeit, derartige Fehler aufzudecken, ist, alle Daten zu vergleichen, kombiniert mit einer Abdeckung des kompletten Programmcodes. Da man nie wissen kann, welche Anweisung falsch konvertiert worden ist, muss jede Anweisung getestet werden – das bedeutet, dass eine wiederholte Durchführung des Tests notwendig ist [9].

Neben der Messung der Testabdeckung ist es ebenso erforderlich, die Richtigkeit der Ausgabedaten und den Prozentsatz an tatsächlich aufgetretenen Fehlern relativ zu jenem der vorausgesagten Fehler zu messen. Diese Information ermöglicht es, festzustellen, wann der Test beendet werden kann. Die definierten Planziele dazu wurden vor Projektstart in der Testvereinbarung verankert (Abbildung 4).

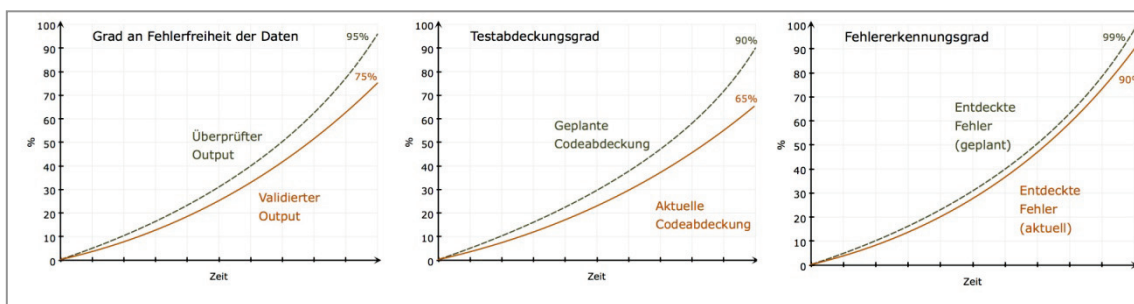


Abbildung 4



ZUSAMMENFASSUNG

Ein migriertes System zu testen erfordert einen anderen Ansatz als ein neu entwickeltes System zu testen. Ein migriertes System einem Software-Test zu unterziehen, betrifft wesentlich stärker den Aspekt der Massenproduktion. Denn eine enorme Anzahl an konvertiertem Programmcode und an Daten muss quasi blindlings innerhalb eines begrenzten Zeitrahmens getestet werden. Um das Ziel der Demonstration einer funktionellen Gleichwertigkeit zwischen Alt- und migriertem System zu erreichen, müssen idealerweise beinahe alle konvertierten Code-Elemente mittels einer großen Datenprobe aus dem Produktivsystem getestet werden. Dieser große Datenbestand und die Unmenge an involvierten Transaktionen, die wiederholt durchgeführt werden müssen, sprechen stark für den Einsatz von Testautomatisierung.

Die Autoren haben die Erfahrung gemacht, dass der Migrationstest hochspezialisierte Tester und hochentwickelte Testwerkzeuge erfordert. Es ist weniger eine Frage der Testeranzahl als eine des Testautomatisierungsgrades. Von ebenso großer Bedeutung ist der Bereich der Testorganisation, da ein Migrationstest sehr sorgfältig bis ins kleinste Detail geplant werden muss. Das macht erfahrene Testmanager notwendig. Da es diese allerdings erst in wenigen Organisationen gibt, benötigt man umso mehr ein spezialisiertes Testteam, um ein Migrationsprojekt erfolgreich abwickeln zu können.

VERFASST VON



Harry M. Sneed ist seit 1969 Magister der Informationswissenschaften der University of Maryland. Seit 1977, als er für das Siemens ITS-Projekt die Rolle des Testmanagers übernommen hat, arbeitet er im Testbereich. Damals entwickelte er die erste europäische Komponententest-Umgebung – PrüfStand – und gründete gemeinsam mit Dr. Ed Miller das erste kommerzielle Testlabor in Budapest. Seit dieser Zeit hat Harry M. Sneed mehr als 20 verschiedene Testwerkzeuge für unterschiedliche Umgebungen entwickelt – von embedded Echtzeitsystemen über integrierte Informationssysteme auf Großrechnern bis hin zu Webapplikationen. Am Beginn seiner Karriere hat er als Testprojektleiter gearbeitet; nun – am Ende seiner langen Karriere – ist er für die ANECON GmbH in Wien in die Rolle eines Software-Testers zurückgekehrt. Parallel zu seiner Projektstätigkeit hat Harry Sneed über 200 technische Artikel und 18 Bücher (davon 4 über das Thema Test) verfasst. Er unterrichtet zudem Software-Entwicklung an der Universität von Regensburg, Software-Wartung an der technischen Hochschule in Linz sowie Software-Messung, Reengineering und Test an den Universitäten von Koblenz und Szeged. 2005 wurde Sneed von der deutschen Gesellschaft für Informatik zum „GI Fellow“ berufen, und übt die Funktion des „general chair“ der internationalen Konferenz für Software-Wartung in Budapest aus. 1996 wurde Sneed vom IEEE für seine Errungenschaften im Bereich des Software Reengineerings ausgezeichnet, und 2008 erhielt er den Stevens Award für seine Pionierarbeit in der Disziplin der Software-Wartung. Sneed ist zertifizierter Tester und aktives Mitglied im österreichischen (ATB) und ungarischen Testing Board.



Seit Anfang 2005 ist **Richard Seidl** als Testspezialist und Testmanager bei der ANECON Software Design und Beratung GmbH tätig. Planung, Konzeption und Durchführung von Testprojekten im Banken- und E-Government-Umfeld bilden den Schwerpunkt seiner Arbeit.

Direkt nach Abschluss seiner Ausbildung zum Ingenieur der Nachrichtentechnik (1999) arbeitete er als freiberuflicher Softwareentwickler, später als Analytiker und Testspezialist bei der Sparkassen Datendienst GmbH in Wien. 2003 übernahm er zusätzlich die Geschäftsführung der SEICON EDV GmbH, die auf begleitende Projektberatung spezialisiert ist.

Die Zertifizierung zum ISTQB® Certified Tester – Full Advanced Level – schloss er Mitte 2006 ab. In diesen Bereich arbeitet er seit 2008 als Trainer. Ende 2007 erlangte er durch die Ausbildung zum IREB Certified Professional for Requirements Engineering die Zertifizierung zum Quality Assurance Management Professional (QAMP).

Gemeinsam mit Harry Sneed und Manfred Baumgartner veröffentlichte Richard Seidl 2006 das Fachbuch „Der Systemtest – Anforderungsbasiertes Testen von Software Systemen“, das Ende 2008 in zweiter Auflage erschien. Auf internationalen Konferenzen ist er mittlerweile ein gefragter Experte, so war er 2009 u.a. als Redner auf der CONQUEST und den Software Quality Days eingeladen.



LITERATUR

- [1] Onoma,A./Tsai,W.-T./ Suganuma, H.: „Regression Testing in an Industrial Environment“, Comm. Of ACM, Vol. 41, Nr. 5, May 1998, S. 81
- [2] DeMarco, T.: Controlling Software Projects – Management, Measurement & Estimation, Yourdon Press, New York, 1982
- [3] Criag, R., Jaskiel, S.: Systematic Software Testing, Artech House Pub., Norwood, MA. , 2002
- [4] Koomen, T., von der Alst, Leo, Broekman, B., Vroon, M.: TMap Next for result-driven Testing, UTN Publishers, Hertogenbosch, NL, 2007
- [5] IEEE: ANSI/IEEE Standard 829 – Standard for Software Test Documentation, ANSI/IEEE Standard 829-1998, Computer Society Press, New York, 1998
- [6] Fewster, M./Graham, D.: Software Test Automation, Addison-Wesley, Harlow, G.B., 1999
- [7] Black, R.: Pragmatic Software Testing, Wiley Publishing, Indianapolis, 2007
- [8] Sneed, H./Baumgartner, M./Seidl,R.: „Der Systemtest“, Hanser Verlag, München-Wien, 2008
- [9] Kann, S.H.: Metrics and Models in Software Quality Engineering, Addison-Wesley, Boston, 2001

