WHITE PAPER | APRIL 2016

Vollständig automatisiertes ETL-Testing: ein Schritt-für-Schritt-Leitfaden

ca technologies

Abschnitt 1

Die wichtige Rolle von ETL für das moderne Unternehmen

Seit der Prozess „Extrahieren, Transformieren, Laden“ (ETL) in der Welt des Data Warehousing und der Business Intelligence eingeführt wurde, hat er sich überall verbreitet. Wie der Name nahelegt, besteht eine ETL-Routine aus drei getrennten Schritten, die häufig parallel ausgeführt werden: Daten werden aus einer oder mehreren Datenquellen extrahiert, sie werden in den erforderlichen Zustand konvertiert, und sie werden in das gewünschte Ziel geladen, meist ein Data Warehouse, einen Data Mart oder eine Datenbank. Eine hoch entwickelte ETL-Routine umfasst häufig auch Fehlerbehandlung, Protokollierungsinfrastruktur und die Umgebung der Routine.[1]

Bis jetzt wurde ETL vor allem verwendet, um häufig große und nicht zusammenhängende Daten für Analysen und Business Intelligence vorzubereiten. Die Nutzung geht jedoch inzwischen über das einfache Verschieben von Daten hinaus: Die Migration von Daten für neue Systeme ist eine zunehmend verbreitete Anwendung. Eine andere ist die Verarbeitung von Integrations-, Sortier- und Join-Vorgängen für Daten.[2]

ETL ist daher ein wichtiges Leistungsmerkmal des heutigen schnelllebigen Entwicklungslebenszyklus, in dem zu jedem Zeitpunkt mehrere Releases und Versionen parallel entwickelt werden. Unternehmen müssen in der Lage sein, ihre Software ständig zu erweitern, zu integrieren und zu innovieren. Dabei müssen für jede Iteration und jedes Release Daten im richtigen Zustand für die Tester und die Entwickler zur Verfügung stehen. Alle diese Daten werden aus denselben Quellen abgerufen, aber sie müssen transformiert werden, um den individuellen Anforderungen der einzelnen Teams zu entsprechen. Dies gilt ganz besonders, wenn ein Unternehmen Agile-Methoden oder die Continuous Delivery erfolgreich implementieren möchte.

Ein gutes Beispiel für die wichtige Rolle von ETL bei der Continuous Delivery fand sich in einer großen internationalen Bank, mit der CA Technologies zusammenarbeitete. Die Bank musste die Kunden, Produkte und Finanzdaten einer gerade übernommenen weiteren Bank in ihre eigene vorhandene Infrastruktur migrieren. Dies bedeutete, dass 80 Einfügungsdateien abgerufen, konvertiert und überprüft werden mussten, bevor sie in die Back End-Systeme der Bank hochgeladen wurden. Außerdem mussten diese Daten für 47 Projekte parallel verfügbar gemacht werden, wobei ihre referenzielle Integrität gewahrt werden musste. In diesem Fall war ETL unerlässlich, um die für eine erfolgreiche Continuous Delivery erforderliche Parallelität zu ermöglichen.

Trotz der gestiegenen Verwendung und Wichtigkeit von ETL spiegelt das ETL-Testing jedoch den allgemeinen Status quo des Testings wider: Es ist zu langsam und erfordert zu viel manuellen Aufwand, und es ermöglicht, dass inakzeptabel viele Fehler bis in die Produktion gelangen. In diesem White Paper werden die Herausforderungen beschrieben, die bei einem typischen Ansatz für das ETL-Testing im Allgemeinen auftreten. Dann wird ein alternativer, umfassend modellbasierter Ansatz vorgestellt, und es wird erörtert, wie er das ETL-Testing weit effizienter, effektiver und systematischer gestalten kann.

Abschnitt 2

Der typische Ansatz für das ETL-Testing und seine häufigen Herausforderungen

Bei der Überprüfung von ETL-Transformationsregeln erstellen Tester im Allgemeinen einen Schattencodesatz, verwenden ihn für die Transformation von Daten und vergleichen dann die tatsächlichen Ergebnisse mit den erwarteten Ergebnissen. Im Allgemeinen werden ETL-Scripts oder SQL-Code manuell zu den Quelldaten kopiert und ausgeführt, und die Ergebnisse werden aufgezeichnet. Dasselbe Script wird dann zu den Zieldaten kopiert, und die Ergebnisse werden aufgezeichnet. Die beiden Ergebnismengen (die tatsächliche und die erwartete) werden nun verglichen, um zu überprüfen, ob die Daten richtig transformiert wurden.

Der Kernaspekt: Komplexität und Testbarkeit

Das Problem, das derartiger manueller Überprüfung zugrunde liegt, besteht darin, dass ETL-Routinen von Haus aus schnell sehr komplex werden. Wenn das Unternehmen wächst und immer unterschiedlichere und umfangreichere Daten erfasst, wachsen die ETL-Regeln, um diese Daten zu bewältigen. Im sogenannten Informationszeitalter erfolgt dieses Wachstum so schnell, dass herkömmliche Testing-Methoden nicht mithalten können. Der schiere Umfang der Informationen, die von datengesteuerten Unternehmen gesammelt werden, ist so schnell angestiegen, dass 90 % der weltweit vorhandenen Daten in den letzten zwei Jahren gesammelt wurden[3], und der Umfang der Daten, die das durchschnittliche Unternehmen sammelt, verdoppelt sich jedes Jahr.[4]

Die Komplexität der Systeme, die entworfen werden, um diese Daten zu erfassen, zu übertragen, zu verarbeiten und zu präsentieren, steigt exponentiell in der Anzahl der hinzugefügten Entscheidungen. Hierzu gehören ETL-Regeln, und es gibt zahlreiche Faktoren, die sich auf die Komplexität der Transformationen auswirken können:

- die Anzahl und Vielfalt der beteiligten Datenquellen, einschließlich relationaler und nicht relationaler Datenbanktypen und Flatfiles
- die Anzahl und Vielfalt der Datenziele
- die Anzahl und Vielfalt einfacher und komplexer Transformationen von einfachen Nachschlagevorgängen bis hin zu aktiven Union-Vorgängen und der Normalisierung
- die Anzahl wiederverwendbarer Transformationen, Codestücke und Joins
- die Anzahl erstellter Tabellen[5]

Alle diese Faktoren werden durch die aktuelle Konzentration auf fast in Echtzeit arbeitende Lösungen und durch die zusätzlichen Komplikationen, die diese verursachen, noch intensiviert.[6]

Die Dokumentation ist nicht hilfreich

Diese wachsende Komplexität hat direkte Auswirkungen auf die Testbarkeit von ETL-Routinen. Dies ist besonders problematisch für das ETL-Testing, da die Transformationsregeln im Allgemeinen in schlechter Dokumentation gespeichert sind, in der die erwarteten Ergebnisse nicht explizit genannt werden. Die Regeln werden oft erst während der Entwicklungsphase entworfen und werden häufig in Fließtextdokumenten oder Kalkulationstabellen gespeichert – oder, was noch schlimmer ist, sie sind möglicherweise nur in den Köpfen der Entwickler und der Tester vorhanden.[7] In diesem Fall existiert keine wirkliche Dokumentation, aus der vertrauenswürdige Testfälle (d. h. der Schattencode) abgeleitet werden können.

Ein Business Intelligence-Team, mit dem wir zusammenarbeiteten, speicherte Anforderungen als Fließtextdokumente und Testfälle in Kalkulationstabellen. Diese statische Dokumentation war ein langer Fließtext, aus dem die logischen Schritte der ETL-Routinen mühsam entnommen werden mussten. Die Dokumente konzentrierten sich auf das optimale Szenario (Happy Path) und enthielten keine negativen Bedingungen, sodass ca. 80 % der möglichen Logik, die in einem durchschnittlichen System getestet werden muss, nicht berücksichtigt waren. Eine derartige unvollständige, uneindeutige Dokumentation bedeutete, dass die Tester keine Möglichkeit hatten, die ETL-Routinen leicht oder präzise zu verstehen.

Zu häufig mussten Tester die Lücken füllen, aber wenn ihnen dabei Fehler unterliefen, schlichen sich Fehler in die ETL-Routinen ein. Dann wurden ungültige Daten an das Ziel kopiert, obwohl der Code und die Testfälle eine plausible Interpretation der Anforderungsdokumentation widerspiegelten.

„Garbage in, Garbage out“ – manuelle Ableitung der Testfälle und der erwarteten Ergebnisse

Enorme 56 % der Fehler, die bis in die Produktion gelangen, können auf Uneindeutigkeiten in der Anforderungsdokumentation zurückgeführt werden.[8] Dies liegt zum Teil daran, dass Testfälle und erwartete Ergebnisse manuell aus der unzureichenden Dokumentation abgeleitet werden: ein sehr zeitaufwendiger Prozess, der im Allgemeinen zu schlechter Testabdeckung führt.

Qualität

Die manuelle Ableitung findet ad hoc und unsystematisch statt und führt im Allgemeinen dazu, dass einfach die Testfälle erstellt werden, die den Testern gerade einfallen. Angesichts der erörterten Komplexität von ETL-Routinen in Kombination mit der verfügbaren schlechten Dokumentation ist es unfair, selbst von den talentiertesten Testern zu erwarten, dass sie jeden Test erstellen, der erforderlich ist, um die möglichen Datenkombinationen zu überprüfen. Wenn beispielsweise ein einfaches System mit 32 Knoten und 62 Kanten auf lineare Weise entworfen wird, gibt es 1.073.741.824 mögliche Pfade durch seine Logik – mehr, als irgendein Mensch auflisten kann.

Die Ad-hoc-Ableitung führt daher dazu, dass einiges deutlich zu wenig und anderes deutlich zu viel getestet wird, wobei nur ein Bruchteil der möglichen Logik einer ETL-Routine getestet wird. Das Negativ-Testing stellt eine besondere Herausforderung dar, und die Testfälle, wie die Dokumentation, konzentrieren sich im Allgemeinen fast nur auf die optimalen Szenarien. Es ist jedoch wichtig, Ausreißer und unerwartete Ergebnisse zu testen, und ETL-Routinen müssen solche fehlerhaften Daten unbedingt ablehnen.

Beispielsweise begnügte sich ein Finanzdienstleister, mit dem CA Technologies zusammenarbeitete, mit elf Testfällen mit nur 16 % Testabdeckung. Dies ist eine ziemlich übliche Zahl, und wir haben bei unseren Audits festgestellt, dass 10–20 % funktionale Testabdeckung die Norm sind. Ein anderes Projekt in diesem Unternehmen wurde 18-mal zu viel getestet, da bei dem Versuch, das System vollständig zu testen, sehr viele Testfälle verwendet wurden; es wurde jedoch keine maximale Abdeckung erreicht. Die Durchführung der 150 zusätzlichen Testfälle, die an einen externen Anbieter ausgegliedert war, kostete 26.000 US-Dollar.[9]

Eine derart schlechte Abdeckung führt dazu, dass Fehler in den Code gelangen, wo ihre Behebung viel Geld und Zeit kostet: Studien haben gezeigt, dass es 40–1.000-mal mehr Ressourcen[10] und 50-mal mehr Zeit[11] kostet, einen Bug beim Testing zu beheben, als wenn er vorher gefunden wird. Was noch schlimmer ist: Manche Bugs werden möglicherweise gar nicht gefunden, sodass ungültige Daten an das Liveziel kopiert werden, wo sie die Integrität des Systems bedrohen können. Auf der Grundlage statischer Dokumentation können Tester außerdem die Abdeckung ihrer Testfälle nicht zuverlässig messen – sie können einfach nicht sicher angeben, einen wie großen oder wie kleinen Teil der jeweiligen Routine sie testen, und sie können den Tests auch keine Prioritäten anhand ihrer Wichtigkeit zuordnen.

Zeit und Aufwand: das Testing kann einfach nicht mithalten

Auch die Erstellung von Testfällen anhand derartiger Dokumentation ist sehr zeit- und arbeitsaufwendig. Im vorigen Beispiel dauerte es 6 Stunden, die 11 Testfälle zu erstellen, während die vielen übermäßigen Tests das Unternehmen sogar noch mehr Zeit kosteten. Zu dieser Zeit, die mit manuellem Testfalldesign verschwendet wird, kommt noch die Zeit hinzu, die dann damit verbracht werden muss, die tatsächlichen und die erwarteten Ergebnisse zu vergleichen.

Die riesigen einzelnen Felder mit den erwarteten Ergebnissen zu vergleichen, kostet sehr viel Zeit, da eine komplexe ETL-Routine umfangreiche Daten produziert und da die Quelldaten häufig in zahlreichen unterschiedlichen Datenbanken und Dateitypen gespeichert sind. Außerdem ist es sehr schwierig, da die transformierten Daten auf mehreren Levels überprüft werden müssen:

- Tester müssen auf Datenvollständigkeit prüfen und sicherstellen, dass die Anzahlen in Datenquelle und -ziel übereinstimmen.
- Die Datenintegrität muss sichergestellt werden, und es muss überprüft werden, ob die Zieldaten mit den Quelldaten konsistent sind.
- Die Transformation muss den Unternehmensregeln entsprechen.
- Die Datenkonsistenz muss garantiert werden, wobei alle unerwarteten Duplikate identifiziert werden müssen.
- Die referenzielle Integrität muss aufrechterhalten werden. Dabei müssen alle verwaisten Datensätze und fehlenden Fremdschlüssel erkannt werden.[12]

Manchmal wird ein Kompromiss eingegangen und nur eine beispielhafte Datenmenge überprüft. Dies kompromittiert jedoch auch die Gründlichkeit des ETL-Testings, sodass die Transformationen weniger zuverlässig sind. Angesichts der Rolle vieler ETL-Routinen in unternehmenskritischen Transaktionen ist ein solcher Kompromiss inakzeptabel. Ebenfalls beeinträchtigt wird die Qualität dadurch, dass manuelle Vergleiche sehr fehleranfällig sind, vor allem, wenn die erwarteten Ergebnisse schlecht definiert sind – oder, was noch schlimmer ist, gar nicht unabhängig vom Schattencode definiert sind, der beim Testing verwendet wird. In diesem Fall nehmen Tester meist an, dass der jeweilige Test bestanden wurde, außer wenn das tatsächliche Ergebnis sehr seltsam ist: Ohne vordefinierte erwartete Ergebnisse nehmen sie meist an, dass das tatsächliche Ergebnis das erwartete Ergebnis ist,[13] und können die Gültigkeit der Daten daher nicht mit Sicherheit bestimmen.

Das Datenproblem

Bis jetzt lag der Schwerpunkt auf den Problemen, die beim Ableiten der Tests (des Schattencodes) auftraten, die für die Überprüfung von ETL-Regeln benötigt wurden. Nach der Ableitung der Testfälle benötigen die Tester jedoch Dummyquelldaten, die das System durchlaufen können. Dies ist eine andere häufige Ursache für Engpässe und Fehler.

Verfügen Sie über alle Daten, die Sie benötigen, um die komplexen ETL-Routinen zu testen?

Für ein effektives ETL-Testing ist es wichtig, über genug „fehlerhafte“ Daten zu verfügen, da im Betrieb unbedingt eine ETL-Regel diese Daten ablehnen und sie im geeigneten Format an den richtigen Anwender senden muss. Wenn diese fehlerhaften Daten nicht abgelehnt werden, führen sie wahrscheinlich zu Fehlern oder sogar zum Zusammenbruch des Systems.

In diesem Kontext gibt es eine Reihe unterschiedlicher Möglichkeiten, „fehlerhafte Daten“ zu definieren. Diese entsprechen den Arten, auf die Tester Daten überprüfen müssen. Es kann sich um Daten handeln, die anhand der Unternehmensregeln niemals akzeptiert werden dürften – beispielsweise negative Werte in einem Onlinewarenkorb, wenn kein Gutschein eingegeben wurde. Es kann sich auch um Daten handeln, die die referenzielle Integrität eines Data Warehouse bedrohen, wie fehlende abhängige oder obligatorische Daten oder Daten, die in der Eingabe fehlen.[14] Die Testdaten, die eine ETL-Überprüfungsregel durchlaufen, müssen daher das gesamte Spektrum ungültiger Daten enthalten, um 100 %ige funktionale Testabdeckung zu erhalten.

Derartige Daten finden sich selten in den Produktionsdatenquellen, die in vielen Unternehmen den Testteams bis heute bereitgestellt werden. Der Grund hierfür ist, dass Produktionsdaten aus vergangenen Alltagsszenarien entnommen werden und daher von Haus aus keine fehlerhaften Daten enthalten. Sie enthalten nicht die unerwarteten Ergebnisse, Ausreißer oder Randbedingungen, die für das ETL-Testing benötigt werden, sondern ihr Schwerpunkt liegt auf optimalen Szenarien. Unsere Audits von Produktionsdaten haben sogar gezeigt, dass eine Abdeckung von 10–20 % die Norm ist. Das Ironische ist: Je besser eine Routine erstellt wurde, umso weniger fehlerhafte Daten lässt sie durch, sodass weniger unterschiedliche Daten für zukünftige Tests der ETL-Regeln vorhanden sind und diese Tests folglich nicht vollständig sind.

Sind die Daten verfügbar, wenn Sie sie brauchen?

Ein weiteres wichtiges Problem für die ETL-Überprüfung ist die Verfügbarkeit von Daten. Quelldaten können aus 50 unterschiedlichen Quellen in einem Unternehmen abgerufen werden. Das Problem für das ETL-Testing und das Testing im Allgemeinen besteht darin, dass es als eine Reihe linearer Phasen betrachtet wird, sodass Testteams gezwungen sind, auf Daten zu warten, während ein anderes Team sie verwendet.

Denken Sie beispielsweise an eine Migrationskette zwischen Banken, bei der Daten von einer Bank abgerufen und mit einem Abstimmungstool für die Systeme der anderen Bank konvertiert werden. In jeder Phase müssen die Daten überprüft werden, um sicherzustellen, dass sie richtig in das Finanzkontrollframework konvertiert wurden, dass die Kontonummer abgerufen wurde, dass Korrektheit im Zeitverlauf bestand usw. Dieser Prozess kann aus mehreren unterschiedlichen Phasen bestehen, von der grundlegenden Eingabe über die Deduplizierung und Vorbereitung bis hin zur Weitergabe und Einbuchung der Daten. Außerdem können mehrere Teams beteiligt sein, darunter sowohl ETL-Teams als auch andere, vor allem Teams, die mit dem Mainframe arbeiten.

Wenn die Daten aus dem gesamten Unternehmen nicht für alle Teams parallel verfügbar sind, wachsen die Verzögerungen, da Teams warten müssen, statt zu arbeiten. Tester schreiben ihren Schattencode und verfügen dann nicht über die notwendigen Quelldaten, um eine ETL-Regel zu überprüfen, da sie von einem anderen Team verwendet werden. Wir haben sogar herausgefunden, dass der durchschnittliche Tester 50 % seiner Zeit damit verbringen kann, auf Daten zu warten, sie zu bearbeiten oder sie zu erstellen. Dies kann enorme 20 % des gesamten SDLC ausmachen.

Was passiert bei Regeländerungen?

Manuell Testfälle und -daten von statischen Anforderungen abzuleiten, ist bei Änderungen kaum reaktionsfähig. ETL-Routinen verändern sich ebenso schnell, wie sich das Unternehmen weiterentwickelt, und der Umfang und die Vielfalt der Daten wachsen mit. Wenn solche ständigen Veränderungen auftreten, kann das ETL-Testing jedoch nicht Schritt halten.

Die wohl wichtigste Ursache für Projektverzögerungen besteht in diesem Fall darin, dass vorhandene Testfälle überprüft und aktualisiert werden müssen, wenn sich die Routinen ändern. Tester können nicht automatisch identifizieren, welche Auswirkungen eine Veränderung auf die statischen Anforderungen und die Testfälle hat. Stattdessen müssen sie jeden vorhandenen Testfall manuell überprüfen, ohne messen zu können, ob die Abdeckung tatsächlich aufrechterhalten wurde.

Für das oben erwähnte Business Intelligence-Team, dessen Anforderungen und Testfälle in Fließtextdokumenten bzw. Kalkulationstabellen gespeichert waren, waren Veränderungen besonders problematisch. Ein Tester benötigte 7,5 Stunden, um einen Satz von Testfällen zu überprüfen und zu aktualisieren, als eine einzelne ETL-Regel geändert wurde. In einem anderen Unternehmen, mit dem wir zusammenarbeiteten, benötigten zwei Tester zwei Tage, um jeden vorhandenen Testfall zu überprüfen, als eine Änderung an den Anforderungen vorgenommen wurde.

Abschnitt 3

Die tragfähige Alternative: vollständig automatisiertes ETL-Testing

Es ist also offensichtlich, dass das ETL-Testing mit manueller Ableitung von Testfällen und manuellen Vergleichen von Ergebnissen nicht mit der Geschwindigkeit Schritt halten kann, mit der Unternehmensanforderungen sich ständig verändern. Im Folgenden wird eine mögliche Strategie beschrieben, mit der die Effizienz und Wirksamkeit des ETL-Testings erhöht werden können. Diese Strategie basiert auf Modellen und Anforderungen und dient dazu, die Testing-Bemühungen früher im Zyklus stattfinden zu lassen („Shift Left“) sowie die Qualität von Anfang an im ETL-Lebenszyklus zu berücksichtigen. Mit einem solchen modellbasierten Ansatz wird die Automatisierung in jeder Phase von Testing und Entwicklung eingeführt, und das ETL-Testing wird vollständig reaktionsfähig für ständige Veränderungen.

1) Beginnen Sie mit einem formalen Modell

Die Einführung des formalen Modeling in das ETL-Testing bietet den grundlegenden Vorteil des „Shift Left" der Testing-Bemühungen. Alle folgenden Assets für Testing und Entwicklung können von der anfänglichen Abbildung einer ETL-Regel auf ein Modell abgeleitet werden. Daher wird das formale Modell zum Grundpfeiler der vollständig automatisierten ETL-Überprüfung.

Das formale Modeling unterstützt Sie jedoch auch bei der Lösung der oben beschriebenen spezifischeren Probleme der Uneindeutigkeit und Unvollständigkeit von Anforderungen. Es hilft Ihnen, die Testbarkeit zu wahren, obwohl die ETL-Regeln immer komplexer werden, sodass Tester die zu testende Logik schnell visuell genau verstehen können. Sie können leicht ermitteln, welche gültigen und ungültigen Daten eingegeben werden sollten, um eine Transformationsregel vollständig zu testen, und welches Ergebnis jeweils erwartet wird.

Beispielsweise wird mit einem Flowchart-Modell ein andernfalls unhandlicher langer Fließtext in handhabbare Stücke aufgeteilt. ETL wird in der Ursache-Wirkung-Logik dieses Modells dargestellt. Diese Zuordnung erzeugt eine Reihe von Wenn-Dann-Aussagen, die mit einer Hierarchie von Prozessen verknüpft sind.[15] Jeder dieser Schritte wird effektiv zu einer Testkomponente, sodass dem Tester genau mitgeteilt wird, was überprüft werden muss. Das Modeling von ETL-Routinen als Flowchart eliminiert daher Uneindeutigkeit aus der Anforderungsdokumentation, um die 56 % Fehler zu vermeiden, die aus dieser Uneindeutigkeit stammen.

Wenn die ETL-Routinen komplexer werden, dient das Flowchart als einzelner Referenzpunkt. Im Gegensatz zu „statischen" Fließtextdokumenten und Diagrammen kann bei diesem Modell zusätzliche Logik leicht hinzugefügt werden. Außerdem ist die Abstrahierung sehr komplizierter Routinen mithilfe der Teilflusstechnologie möglich. Dies erhöht die Testbarkeit. Komponenten auf niedrigerem Level können in Masterabläufe eingebettet werden, sodass die zahlreichen Routinen, aus denen ein hoch komplexer Satz von ETL-Regeln besteht, zu einem einzelnen, visuellen Diagramm konsolidiert werden können.

Das Flowchart Modeling reduziert nicht nur Uneindeutigkeiten, sondern ermöglicht auch die Verminderung von Unvollständigkeit. Es zwingt den Modellierer, über Beschränkungen, negative Bedingungen, Einschränkungen und Randbedingungen nachzudenken und zu fragen: „Was passiert, wenn diese Ursache oder dieser Auslöser nicht vorhanden ist?" Der Modellierer muss also systematisch negative Pfade konstruieren und das Negativ-Testing berücksichtigen, das den Großteil der ETL-Überprüfung ausmachen sollte. Außerdem können Algorithmen angewendet werden, die auf Vollständigkeit überprüfen, da es sich bei dem formalen Modell um ein mathematisch präzises Diagramm der ETL-Regel handelt.

Dies eliminiert fehlende Logik wie das „Dangling Else". Daher können Testfälle abgeleitet werden, die 100 % der möglichen Datenkombinationen abdecken (beachten Sie jedoch, dass es fast immer mehr Kombinationen gibt, als realistischerweise als Tests durchgeführt werden können; daher werden weiter unten Optimierungstechnologien erörtert). Ein weiterer wesentlicher Vorteil besteht darin, dass die erwarteten Ergebnisse in dem Modell definiert werden können, unabhängig von den Testfällen. In anderen Worten kann der Anwender mit einem Flowchart das Modell so definieren, dass es die Randeingaben einschließt, und die zugehörigen erwarteten Ergebnisse an unterschiedliche Endpunkte im Modell weitergeben. Damit wird klar definiert, was eine Überprüfungsregel akzeptieren und was sie ablehnen sollte, damit Tester nicht fälschlich annehmen, dass Tests bestanden wurden, wenn das erwartete Ergebnis nicht explizit vorliegt.

Beachten Sie, dass ein modellbasierter Testing-Ansatz für die ETL-Überprüfung auch eingeführt werden kann, ohne unternehmensweit einen anforderungsbasierten Ansatz für das Testing und die Entwicklung vollständig einzuführen. Es ist keine vollständige Veränderung notwendig, und unserer Erfahrung nach dauert es nur 90 Minuten, eine ETL-Routine als „aktives" Flowchart zu modellieren. Dieses Modell kann dann vom ETL- oder Testteam ausschließlich mit dem Ziel verwendet werden, das Testing durchzuführen und dann diese Routine selbst noch einmal zu testen.

2) Leiten Sie Testfälle automatisch aus dem Flowchart-Modell ab

Mit der Einführung des modellbasierten Testings können Sie eines der wichtigsten manuellen Elemente des ETL-Testings automatisieren: das Testfalldesign. Tester müssen keinen Schattencode mehr schreiben und nicht mehr manuell SQL-Code aus der Quell- in die Zieldatenbank kopieren. Stattdessen werden die Pfade durch das Flowchart zu den Testfällen, anhand derer Daten die Transformationsregeln durchlaufen können. Diese können auf eine Weise systematisch abgeleitet werden, die nicht möglich ist, wenn Code anhand statischer Anforderungen geschrieben wird.

Diese automatische Ableitung ist möglich, weil die gesamte funktionale Logik eines Systems über das Flowchart gelegt werden kann. Dann können automatisierte mathematische Algorithmen angewendet werden, um jeden möglichen Pfad durch das Modell zu identifizieren. Dabei werden Testfälle erzeugt, die jede Kombination von Eingaben und Ausgaben abdecken (dies kann mithilfe der Ursache-Wirkung-Analyse oder der homotopischen Analyse durchgeführt werden).

Da die Testfälle direkt mit dem Modell verknüpft sind, decken sie die gesamte darin definierte Logik ab. Daher bieten sie 100 %ige funktionale Abdeckung, sodass die Verwendung eines Flowchart-Modells zur Erstellung einer vollständigen Dokumentation äquivalent dazu ist, auf das vollständige Testing einer ETL-Routine hinzuarbeiten. Ein weiterer Vorteil dieser Methode besteht darin, dass das Testing messbar wird. Da jeder mögliche Testfall abgeleitet werden kann, können Tester genau ermitteln, wie viel funktionale Abdeckung ein gegebener Satz von Testfällen bietet.

Optimierung: mit weniger Tests mehr testen

Dann können automatisierte Optimierungsalgorithmen angewendet werden, um die Anzahl der Testfälle auf das absolute Minimum zu reduzieren und zugleich maximale funktionale Abdeckung beizubehalten. Diese kombinatorischen Techniken werden durch die logische Struktur des Flowcharts ermöglicht, wobei ein einzelner Schritt in der Ursache-Wirkung-Logik (ein Block im Flowchart bzw. eine Testkomponente) in mehreren Pfaden durch den Ablauf enthalten sein kann. Um die ETL-Routine vollständig zu testen, muss nun jeder einzelne Block (Operator) getestet werden, wobei eine von mehreren vorhandenen Optimierungstechniken verwendet wird (alle Kanten, alle Knoten, alle eingehenden/ausgehenden Kanten, alle Paare). Beispielsweise sind möglicherweise drei Testfälle ausreichend, um die Logik in fünf Pfaden vollständig zu testen.

Beim oben erwähnten Finanzdienstleister erwies sich dies als extrem wertvoll, da es dazu führte, dass übermäßiges Testen reduziert wurde, Testzyklen verkürzt wurden und die Qualität des Testings erhöht wurde. Einschließlich des Zeitaufwands für das Modeling des Flowcharts dauerte es beispielsweise 40 Minuten, 19 Testfälle mit 95 % Abdeckung zu erstellen – im Gegensatz zu den zuvor verwendeten über 150 Testfällen mit 80 % Abdeckung und 18-fach übermäßigem Testing. In einem anderen Projekt dauerte es zwei Stunden, 17 Testfälle mit 100 % Abdeckung zu erstellen. Dies war eine drastische Verbesserung gegenüber der 16 % Abdeckung, die zuvor in sechs Stunden erreicht worden war.

3) Erstellen Sie automatisch die erforderlichen Daten für die Durchführung der Tests

Nach der Erstellung der Testfälle benötigen Tester Daten, die 100 % der möglichen Tests bei ihrer Durchführung abdecken können. Derartige Daten können ebenfalls direkt aus dem Modell abgeleitet werden. Sie können automatisch erstellt oder aus mehreren Quellen gleichzeitig abgerufen werden.

Eine Engine für die Erzeugung synthetischer Daten wie CA Test Data Manager bietet mehrere Möglichkeiten, die erforderlichen Daten zu erstellen, wenn das ETL-Testing als modellbasiertes Testing umgesetzt wird. Der Grund hierfür ist, dass nicht nur funktionale Logik über ein Flowchart gelegt werden kann, sondern auch alle Daten eines Systems. In anderen Worten: Beim Modeling der ETL-Regeln können Sie Ausgabenamen, Variablen und Standardwerte für jeden einzelnen Knoten definieren. Bei der Erstellung der Testfälle können die Daten, die für ihre Ausführung erforderlich sind, sowie die relevanten erwarteten Ergebnisse automatisch aus Standardwerten erzeugt werden.

Alternativ können Systemdaten mit CA Agile Requirements Designer (früher Agile Designer von Grid-Tools) schnell mithilfe des Data Painter-Tools erstellt werden. Es bietet eine umfassende Liste kombinierbarer Datenerzeugungsfunktionen, Seed-Tabellen, Systemvariablen und Standardvariablen. Diese können verwendet werden, um Daten zu erstellen, die jedes mögliche Szenario abdecken, einschließlich fehlerhafter Daten und negativer Pfade. Da jeder Pfad einfach einem weiteren Datenpunkt entspricht, können die Daten, die erforderlich sind, um systematisch zu testen, ob eine ETL-Routine ungültige Daten ablehnen kann, ganz und gar synthetisch erstellt werden.

Und schließlich können vorhandene Daten mithilfe des automatisierten Data Mining innerhalb von Minuten in mehreren Back End-Systemen gefunden werden. Hierbei werden statistische Analysen verwendet, um Muster in Datenbanken zu erkennen, sodass Gruppen von Mustern, Abhängigkeiten oder ungewöhnlichen Datensätzen extrahiert werden können.

4) Provisionieren Sie die Daten innerhalb von Minuten, passend zu den jeweiligen Tests

Im Allgemeinen sollte eine Kombination aus vorhandenen Daten, Produktionsdaten und synthetischen Daten bevorzugt werden, wobei die synthetische Erzeugung verwendet wird, um die Abdeckung auf 100 % zu ergänzen. Für ein effizientes ETL-Testing ist es unerlässlich, dass die Ursprungskopie der Daten intelligent gespeichert wird, sodass dieselben Datenmengen parallel angefordert, geklont und bereitgestellt werden können. Dies eliminiert die Verzögerungen, die durch Dateneinschränkungen verursacht werden.

Der erste Schritt zum intelligenten Daten-Storage besteht darin, einen Test Mart zu erstellen, in dem Daten zu spezifischen Tests zugeordnet werden. Jedem Test werden exakte Daten zugewiesen. Die Daten werden definierten, stabilen Kriterien zugeordnet, nicht spezifischen Schlüsseln. Durch die Zuordnung von Testdaten kann die Zeit eingespart werden, die sonst damit verbracht würde, in umfangreichen Produktionsdatenquellen nach Daten zu suchen, da Daten automatisch aus dem Test Data Warehouse abgerufen oder innerhalb von Minuten per Data Mining aus mehreren Back End-Systemen bezogen werden können.

Das Test Data Warehouse dient als zentrale Bibliothek, in der Daten als wiederverwendbare Assets gespeichert werden, zusammen mit den zugehörigen Abfragen, mit denen sie extrahiert werden. Nun können Datenpools innerhalb von Minuten angefordert und erhalten werden. Sie sind mit den richtigen Testfällen und den erwarteten Ergebnissen verknüpft. Je mehr Tests ausgeführt werden, umso größer wird diese Bibliothek, bis praktisch jede Datenanforderung in einem Bruchteil der Zeit durchgeführt werden kann.

Sehr wichtig ist, dass Daten zugleich an mehrere Systeme übertragen werden können und bei der Provisionierung geklont werden. Dies bedeutet, dass Datenmengen aus zahlreichen Quelldatenbanken für mehrere Teams parallel zur Verfügung stehen. Das ETL-Testing ist kein linearer Prozess mehr, und die langen Verzögerungen, die aus Dateneinschränkungen entstehen, werden vermieden. Die ursprünglichen Daten können gepflegt werden, wenn Änderungen am Modell vorgenommen werden, sodass Teams parallel mit mehreren Releases und Versionen arbeiten können. Diese „Versionssteuerung“ bedeutet auch, dass Änderungen, die an den ETL-Routinen vorgenommen werden, automatisch in den Daten widergespiegelt werden, sodass Testteams die aktuellen Daten erhalten, die sie benötigen, um Transformationen rigoros zu testen.

5) Nutzen Sie die Daten für die Ausführung anhand der Regeln, und vergleichen Sie die Ergebnisse automatisch

Sobald Tester über die Tests verfügen, die erforderlich sind, um eine ETL-Routine vollständig zu testen, sowie über die für ihre Durchführung erforderlichen Daten, muss auch die Überprüfung automatisiert werden, damit das ETL-Testing mit den veränderlichen Anforderungen Schritt halten kann.

Verwenden einer Datenorchestrierungs-Engine

Eine Möglichkeit hierfür besteht in der Verwendung einer Testautomatisierungs-Engine. CA Technologies bietet den Vorteil, dass diese zugleich als Datenorchestrierungs-Engine verwendet werden kann. Dies bedeutet, dass die Daten spezifischen Tests und erwarteten Ergebnissen zugeordnet sind, abgerufen werden können und eine Überprüfungsregel durchlaufen können. Dies ist die „datengesteuerte Automatisierung“, bei der beispielsweise ein Testrahmen, der in einer Datenorchestrierungs-Engine erstellt wurde, jede Zeile einer XML-Datei, die im Flowchart definiert ist, als Test ausführen kann.

Das Ergebnis ist dann Bestehen oder Nichtbestehen, basierend auf den erwarteten Ergebnissen, die im Modell definiert sind. Dies automatisiert sowohl die Durchführung von Tests als auch den Vergleich der tatsächlichen Ergebnisse mit den erwarteten. Tester müssen Scripts nicht mehr manuell vom Datenziel zur Datenquelle kopieren. Außerdem vermeiden sie den mühsamen, fehleranfälligen Prozess, jedes einzelne durch die Transformation erstellte Feld zu vergleichen.

Verwendung von CA Test Data Manager

Alternativ kann, nachdem die erwarteten Ergebnisse definiert wurden, CA Test Data Manager verwendet werden, um auf dieser Grundlage automatisch Berichte zum Bestehen/Nichtbestehen zu erstellen. Damit werden die andernfalls manuellen Datenvergleiche automatisiert, aber es muss weiterhin SQL-Code von der Quelle zum Ziel kopiert werden.

Zuerst werden mithilfe der oben beschriebenen Techniken synthetische Daten im Quellsystem definiert. Dann kann ein Datenpool erstellt werden, um den ETL-Prozess zu simulieren, mit dem Daten von der Quelle zum Ziel kopiert werden. Eine gültige Datenmenge kann kopiert werden, um zu testen, ob sie akzeptiert wird, und eine fehlerhafte Datenmenge, um sicherzustellen, dass ungültige Daten abgelehnt werden. Indem eine weitere Tabelle erstellt wird, in der die Testbedingungen und -ergebnisse gespeichert werden, und eine auf dem Datenpool basierende Tabelle mit Testfällen und Datenbedingungen, können die erwarteten Ergebnisse automatisch mit den tatsächlichen Ergebnissen verglichen werden. Alle fehlenden oder fehlerhaften Daten können dann auf einen Blick identifiziert werden.

6) Implementieren Sie Änderungen automatisch

Einer der größten Vorteile des modellbasierten Testings für die ETL-Überprüfung besteht in der Möglichkeit, auf schnelle Veränderungen zu reagieren. Da die Testfälle, Daten und Anforderungen so eng miteinander verknüpft sind, kann eine am Modell vorgenommene Änderung automatisch in den Testfällen und zugehörigen Daten widergespiegelt werden. ETL-Routinen werden schnell immer komplexer. Diese Nachverfolgbarkeit bedeutet, dass das Testing mithalten kann und keine Engpässe in der Pipeline der Anwendungsbereitstellung erzeugt.

Mit dem Flowchart Modeling wird die Implementierung einer Änderung so schnell und einfach wie das Hinzufügen eines neuen Blocks zum Flowchart. Dann können Algorithmen angewendet werden, die die Vollständigkeit überprüfen, um das Modell zu überprüfen und sicherzustellen, dass die gesamte Logik ordnungsgemäß zu einem vollständigen Ablauf verbunden wurde. Wenn Sie CA Agile Requirements Designer verwenden, können Sie mit Path Impact Analyzer die Auswirkungen von Änderungen auf die Pfade durch das Flowchart genauer identifizieren. Die betroffenen Testfälle können dann automatisch entfernt oder repariert werden. Dabei werden ggf. alle neuen Tests, die erforderlich sind, um 100 %ige funktionale Abdeckung zu wahren, automatisch erzeugt.

Dies vermeidet die Zeitverschwendung für manuelle Überprüfungen und Aktualisierungen von Tests. Zugleich wurde gezeigt, dass die automatisierte Deduplizierung Testzyklen um 30 % verkürzt. Das oben erwähnte Business Intelligence-Team konnte mit modellbasiertem Testing hohe Zeitersparnisse beim ETL-Prozess realisieren. Zuvor hatte es 7,5 Stunden gedauert, Testfälle zu überprüfen und zu aktualisieren, wenn eine einzelne ETL-Regel geändert wurde. Mit CA Agile Requirements Designer dauerte es nur zwei Minuten. Außerdem waren von allen Testfällen tatsächlich nur drei betroffen und mussten daher aktualisiert werden.

Wie beschrieben, bedeutet die Versionssteuerung von Daten auch, dass Tester die aktuellen Daten erhalten, die sie benötigen, um eine ETL-Routine vollständig zu testen, auch nachdem sie geändert wurde. Da Testdaten zum Modell nachverfolgt werden können, werden Änderungen an Anforderungen automatisch in den relevanten Datenmengen widergespiegelt. Daten sind für unterschiedliche Releases und Versionen parallel verfügbar. Die notwendigen Daten für effiziente Regressionstests werden dauerhaft im Test Data Warehouse gespeichert. Auch interessante oder seltene fehlerhafte Daten können dauerhaft gespeichert werden, damit sie nicht durch ein anderes Team zerstört werden oder bei einer Datenaktualisierung verloren gehen.

Abschnitt 4

Zusammenfassung

Jedes Unternehmen, das eine Continuous Delivery hochwertiger Software anstrebt, benötigt eine stärkere Automatisierung des ETL-Testings. Vor allem die ETL-Überprüfung erfordert bisher noch hohen manuellen Aufwand, vom Schreiben des Schattencodes über statische Anforderungen bis hin zur Beschaffung der erforderlichen Daten und zum Vergleich der Ergebnisse. Modellbasiertes Testing und intelligentes Test Data Management können verwendet werden, um alle diese Aufgaben zu automatisieren und es zugleich zahlreichen Teams zu ermöglichen, parallel anhand derselben Datenquellen zu arbeiten.

Das modellbasierte Testing erreicht einen „Shift Left" für das ETL-Testing: Der Großteil der Arbeit wird auf die Designphase konzentriert. Danach kann jedes für das ETL-Testing erforderliche Asset automatisch in einem Bruchteil der Zeit abgeleitet werden. Testfälle, die 100 %ige funktionale Abdeckung bieten, können automatisch erzeugt werden und werden mit unabhängig definierten erwarteten Ergebnissen verknüpft. Jeder Test wird dann genau den Quelldaten zugeordnet, die für seine Ausführung erforderlich sind. Wenn diese im Test Data Warehouse gespeichert sind, können sie zahlreichen Teams parallel provisioniert werden.

Aufgrund der engen Verknüpfung, die zwischen Tests, Daten und den Anforderungen erstellt wird, können die Tests einer ETL-Routine, die nach einer Veränderung erneut notwendig sind, schnell durchgeführt werden. Die Zeit, die anfangs für die Erstellung des Modells aufgewendet wird, wird daher schnell durch die Zeit aufgewogen, die gespart wird, weil Tests nicht mehr manuell erstellt, aktualisiert und erneut ausgeführt werden müssen. Die modellbasierte Erzeugung bietet auch den wesentlichen Vorteil der Wiederverwendbarkeit. Dabei können Testkomponenten als gemeinsam nutzbare Assets im Test Data Warehouse gespeichert sowie mit Daten und erwarteten Ergebnissen verknüpft werden. Je mehr Tests ausgeführt werden, umso größer wird die Bibliothek, bis das Testing neuer oder aktualisierter ETL-Regeln so schnell und einfach ist wie die Auswahl aus vorhandenen Komponenten.

Das ETL-Testing führt nicht mehr zu Engpässen bei der Anwendungsbereitstellung und kann mit dem Wachstum datengesteuerter Unternehmen Schritt halten. Die Testbarkeit immer komplexer werdender Routinen wird gewahrt, sodass das Testing mit der Vielfalt und dem Umfang der erfassten Daten umgehen kann und die Continuous Delivery hochwertiger Anwendungen nicht verhindert.

Kontaktieren Sie CA Technologies unter ca.com/de.

CA Technologies (NASDAQ: CA) entwickelt Software, die Unternehmen bei der Umstellung auf die Application Economy unterstützt. Software steht in allen Branchen und in allen Unternehmen im Mittelpunkt. Ob Planung, Entwicklung, Management oder Security – CA Technologies arbeitet weltweit mit Unternehmen zusammen, um die Art, wie wir leben, Transaktionen abwickeln und kommunizieren in mobilen, privaten und öffentlichen Cloud-Umgebungen oder in verteilten Systemen und Mainframe-Umgebungen neu zu gestalten. Weitere Informationen zu unseren Customer Success-Programmen finden Sie unter **ca.com/customer-success**. Weitere Informationen finden Sie unter **ca.com/de**.

1 Jean-Pierre Dijcks, *Why does it take forever to build ETL processes?*, abgerufen am 24.07.2015 von **https://blogs.oracle.com/datawarehousing/entry/why_does_it_take_forever_to_bu**

2 Alan R. Earls, *The State of ETL: Extract, Transform and Load Technology*, abgerufen am 21.07.2015 von **http://data-informed.com/the-state-of-etl-extract-transform-and-load-technology/**

3 IBM, abgerufen am 20.07.2015 von **http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html**

4 Jacek Becla und Daniel L. Wang, *Lessons Learned from managing a Petabyte*, S. 4, abgerufen am 19.02.2015 von **http://www.slac.stanford.edu/BFROOT/www/Public/Computing/Databases/proceedings/**

5 **http://etlcode.com/index.php/utility/etl_complexity_calculator**

6 Jean-Pierre Dijcks, *Why does it take forever to build ETL processes?*, abgerufen am 24.07.2015 von **https://blogs.oracle.com/datawarehousing/entry/why_does_it_take_forever_to_bu**

7 ETL Guru, *ETL Startegy to store data validation rules*, abgerufen am 22.07.2015 von **http://etlguru.com/?p=22**

8 Bender RBT, *Requirements Based Testing Process Overview*, abgerufen am 05.03.2015 von **http://benderrbt.com/Bender-Requirements%20Based%20Testing%20Process%20Overview.pdf**

9 Huw Price, *Test Case Calamity*, abgerufen am 21.07.2015 von **https://communities.ca.com/community/ca-agile-requirements-designer/blog/2016/02/24/test-case-calamity**

10 Bender RBT, *Requirements Based Testing Process Overview*

11 Software Testing Class, *Why testing should start early in software development life cycle?*, abgerufen am 06.03.2015 von **http://www.softwaretestingclass.com/why-testing-should-start-early-in-software-development-life-cycle/**

12 datagaps, *ETL Testing Challenges*, abgerufen am 24.07.2015 von **http://www.datagaps.com/etl-testing-challenges**

13 Robin F. Goldsmith, *Four Tips for Effective Software Testing*, abgerufen am 20.07.2015 von **http://searchsoftwarequality.techtarget.com/photostory/4500248704/Four-tips-for-effective-software-testing/2/Define-expected-software-testing-results-independently**

14 Jagdish Malani, *ETL: How to handle bad data*, abgerufen am 24.07.2015 von **http://blog.aditi.com/data/etl-how-to-handle-bad-data/**

15 Philip Howard, *Automated test case generation*, S. 6, abgerufen am 22.07.2015 von **http://www.agile-designer.com/wpcms/wp-content/uploads/2014/10/00002233-Automated-test-case-generation.pdf**

CS200-196717_0416