Andre Fuss
(andre.fuss@sogeti.de)
ist seit 2011 als Berater für Softwarequalität und Softwaretest für die Sogeti Deutschland GmbH tätig. Seine Branchen-Kenntnisse sind weit gefächert. Seine Schwerpunkte liegen unter anderem im Testdatenmanagement.

Ein Praxisbeispiel: Testdatenmanagement in agilen Projekten

In heutigen Entwicklungsprojekten hantieren sowohl Entwickler als auch Tester bei ihrer Arbeit nahezu täglich mit Testdaten. Qualitativ hochwertige und realistische Testdaten ermöglichen die Durchführung valider Tests. Je höher die Qualität von Testdaten ist, umso besser und nachvollziehbarer werden die Ergebnisse der durchgeführten Tests ausfallen. In agilen Projekten ist es eine große Herausforderung, die Testdaten in jedem Sprint stets datenbankarchitekturkonform zu halten, da sich die Datenbankstruktur stetig ändert beziehungsweise sich zu jedem Sprint erweitert. Diese Arbeit kann immer wieder aus dem Ruder laufen, nicht nur, weil die Daten unter anderem stetig komplexer werden.

Ein effektives Testdatenmanagement benötigt definierte Prozesse und angemessene Werkzeuge zur Verbesserung der Testabdeckung, zur Optimierung des Testaufwands und zur Senkung der Kosten der Qualitätssicherung. Testdatenmanagement ergänzt das Testmanagement durch daten- und testdatenspezifische Aspekte. Darüber hinaus optimiert ein Testdatenmanagement die Erzeugung und Bereitstellung von Testdaten durch die Berücksichtigung datenspezifischer Besonderheiten. In diesem Artikel illustrieren wir anhand eines Beispiels das Einsparpotenzial durch systematisches Testdatenmanagement.

Ausgangssituation

Das hier vorgestellte Praxisbeispiel ist ein Alt-System aus dem Public Sektor in Form einer Sammlung mehrerer Programme, die keine Verbindung beziehungsweise Schnittstelle zueinander besitzen. Es findet kein synchronisierter Datenaustausch zwischen den einzelnen Anwendungen statt. Jegliche Daten müssen per Hand in die anderen Systeme eingegeben werden. Anwender haben alle nötigen Programme geöffnet, die alle bedient werden müssen (je nach Workflow).

Dies soll sich mit einer neuen Software ändern, auch um die Benutzerakzeptanz und den Bedienkomfort zu erhöhen. Das Einsparpotenzial bei den unterstützten Arbeitsabläufen ist enorm. Durch die Ablösung der Alt-Verfahren ist die Zielsetzung des

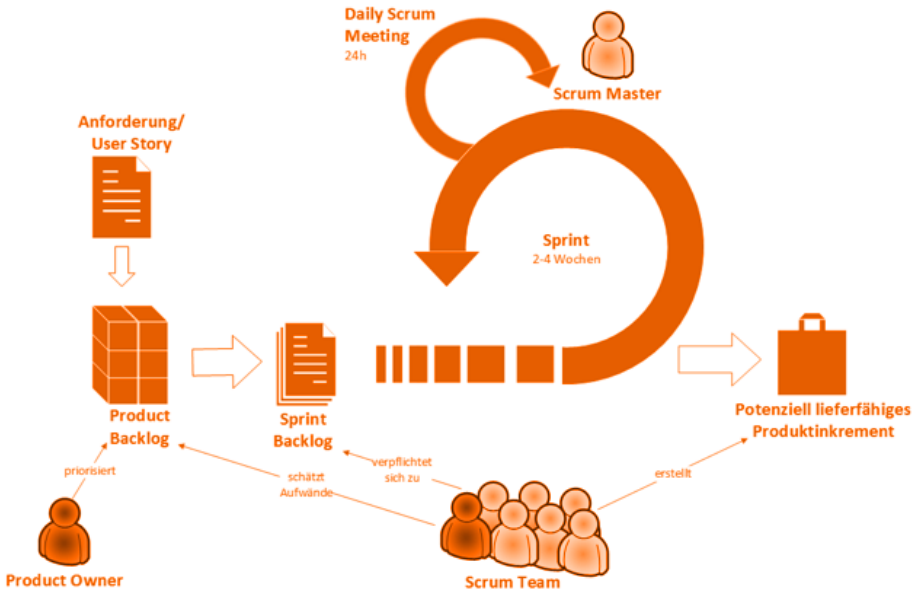

Abb. 1: Vorgehensmodell Entwicklung (Scrum)
(Quelle: http://www.brightsolutions.de/blog/wie-scrum-ihre-projekte-verbessert*).*

```
For Count = 1 To 21500
    Worksheets("Tabelle1").Cells(Count, 1).Value = ""
Next Count

Count = 1

For Y = 2 To 21500

    id = Worksheets("TABELLENBLATT").Cells(Y, 1).Value
    fk_t_Variable = Worksheets("TABELLENBLATT").Cells(Y, 2).Value
    tag = Worksheets("TABELLENBLATT").Cells(Y, 3).Value
    fehltag = Worksheets("TABELLENBLATT").Cells(Y, 4).Value
    fehlstunden = Worksheets("TABELLENBLATT").Cells(Y, 5).Value
    kommentar = Worksheets("TABELLENBLATT").Cells(Y, 6).Value

    sql = "INSERT INTO db.tabelle1 (id, fk_t_Variable, tag, fehltag, fehlstunden, kommentar) VALUES (" + id + ", " + fk_t_Variable + ", '" + tag + "', " _
          + "" + fehltag + ", " + fehlstunden + ", '" + kommentar + "');"

    Worksheets("Tabelle1").Cells(Count, 1).Value = sql

    Count = Count + 1

Next Y

MsgBox ("Die SQL-INSERT-Scripte wurden erfolgreich generiert!")
```

Abb. 2: Entwickelte VBA-Funktion, um Daten auszulesen und SQL-INSERT-Skripte zu erzeugen.

Projekts, den Benutzern eine einheitliche, zukunftsfähige und bedienerfreundliche Software zur Verfügung zu stellen. Diese neue Software muss sich als ein Bestandteil in eine Gesamtlösung mit weiteren IT-Verfahren einbetten. Sie soll den zukünftigen Anwendern und den zuständigen Organisationseinheiten ein leistungsfähiges Instrument bieten.

Aufseiten des Kunden soll die Software in verschiedenen Steuerungs- und Arbeitsbereichen einsetzbar sein. Eine große Rolle spielt dabei die Vereinfachung übergreifender Abstimmungsprozesse. Es gilt ein professionelles Testdatenmanagement (TDM) zu installieren. Die gesetzlichen und regulatorischen Anforderungen im Rahmen des Testdatenmanagements müssen zwingend eingehalten werden.

Die Architektur der Datenbanken des Alt- und des Neu-Systems unterscheiden sich stark voneinander. Bei Fertigstellung der neuen Software müssen die Echtdaten korrekt und ohne Datenverluste in das neue System überführt werden. Dazu bedarf es komplexer Testdaten, die mit der neuen Software mitwachsen, sodass am Ende die Produktivdaten in die neue Datenbankstruktur überführt werden können.

Umsetzung

Für effiziente Lösungen hinsichtlich mehr Wirtschaftlichkeit und höherer Qualität unter anderem beim Testen bietet sich ein gut aufgestelltes Testdatenmanagement an für die Erstellung und Pflege beispielsweise einer Testdatenbasis.

Um eine Migration von Bestandsdaten des Alt-Systems inklusive bestehender Relationen der einzelnen Datensätze mit hoher Qualität durchzuführen, ist es obligatorisch, dass die wachsenden Testdaten in die finalen Datenbankstruktur vollständig transformiert werden können, ansonsten wird eine anschließende Übertragung von Echtdaten in das neue System nicht möglich sein.

Zum Zeitpunkt der Softwareentwicklung des neuen Systems existiert die neue Datenbankstruktur der Systemlandschaft bereits in der Planung. Die neue Architektur auf der Entwicklungs-, Test- und Schulungsumgebung wächst von Sprint zu Sprint (siehe Abbildung 1).

Eine Entwicklung beispielsweise eines Testdatengenerators, der in der Lage wäre, entsprechende Testdaten je nach gewünschter Art zu erzeugen, ist meist sehr kosten- und zeitintensiv. Nach Abstimmung mit den Stakeholdern ist in diesem Projekt ein solches Werkzeug nicht notwendig. Eine kostengünstigere Lösung mit Microsoft Excel in Verbindung mit der Skriptsprache VBA (Visual Basic for Applications) verspricht eine ideale, simple und schnelle Lösung, um die benötigten Testdaten zu erzeugen.

Welche Testdaten zu welchem Zeitpunkt in welcher Umgebung benötigt werden, hängt unter anderem davon ab, wie und wo die Testdaten beschafft werden können. Eine Analyse ist notwendig, um realistische Testdaten zu erzeugen. Eine professionelle Identifikation von Testdatenquellen ist un-

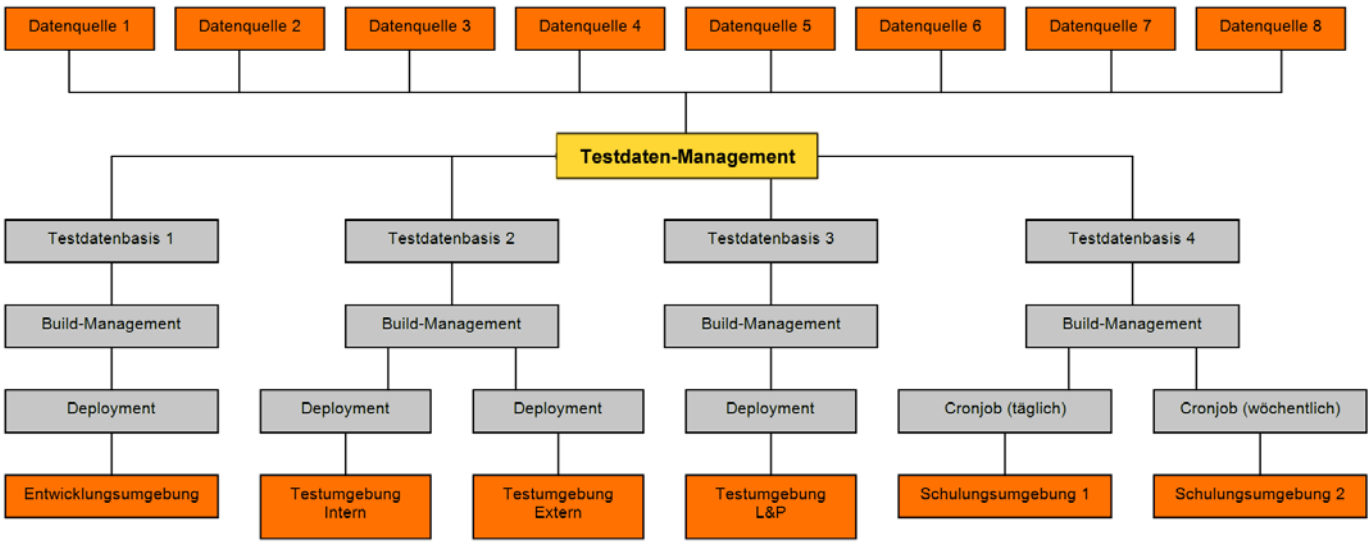

Abb. 3: Ablaufdiagramm des Vorgehens für die Überführung in die verschiedenen Umgebungen.

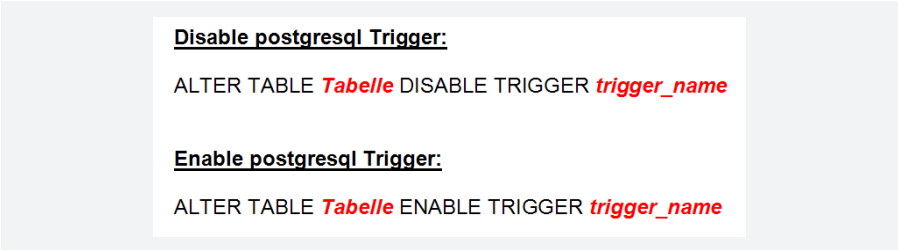

Abb. 4: Beispiel: Trigger ein- und ausschalten.

abdingbar. In den meisten Projekten werden sehr viele und durchaus komplexe Testdaten benötigt. In diesem Praxisbeispiel wird eine immer wieder einsetzbare Testdatenbasis erstellt und diese stetig weiterentwickelt beziehungsweise angepasst.

Die Testdaten werden nach einer Analyse in Excel-Tabellenblätter geschrieben und mittels Excel-Makros und entwickelten/geschriebenen Funktionen als INSERT-SQL-Queries erzeugt beziehungsweise umgewandelt (siehe Abbildung 2). Da innerhalb der Testdaten viele Fremdschlüssel berücksichtigt werden müssen, sind die SQL-Strings relativ komplex. Eine professionelle Erzeugungslogik der Skripte, welche Tabellen mit welchen Daten wann befüllt werden müssen und die Relationen ihre Validität behalten, spielt dabei eine große Rolle.

Welche Testdaten wann und in welcher Form und in welchem Sprint benötigt werden, wird dem Tool „Jira“ entnommen. Die einzelnen User-Stories in Jira geben vor, welche Daten zum jeweiligen Sprint erstellt werden müssen, um die Features erfolgreich zu testen.

Eine große und brauchbare Testdatenbasis muss für jeweilige Umgebungen in jeder Iteration oder zu abgestimmten Zeitpunkten individuell bereitgestellt werden. Es empfiehlt sich daher, hierfür ein geeignetes Werkzeug für ein automatisches Deployment in die einzelnen Umgebungen einzusetzen (z. B. Jenkins).

Des Weiteren müssen bei den Echt-Daten eine Reihe von Datenschutzrichtlinien eingehalten werden, damit unter keinen Umständen Rückschlüsse auf etwaige Personen gezogen werden können. Für die Migration von Alt-Daten werden die Daten entsprechend aufbereitet und für die Tests anonymisiert.

Die benötigten Testdaten sollen in jedem Fall für zum Beispiel die Benutzerakzeptanz realitätsnah sein. Eine kontinuierliche Bereitstellung und Pflege beziehungsweise Anpassung der Testdaten ist obligatorisch. Abbildung 3 zeigt, dass Daten aus unterschiedlichen Quellen in die verschiedenen Umgebungen zu unterschiedlichen Zeitpunkten und teilweise mit einem Cronjob automatisiert eingespielt werden.

Die kontinuierliche Bereitstellung der Information über den aktuellen Stand der Testdatenbasen für die verschiedenen Umgebungen gegenüber des Product Owner hat eine hohe Priorität. Eine schnelle Reaktion auf neue oder veränderte Testanforderungen muss in diesem agilen Projekt eingehalten werden, damit die Testdaten immer valide und brauchbar sind.

Einige Constraints (Fremdschlüsselbeziehungen bzw. Relationen) verweisen auf die eigene Tabelle, daher müssen die betroffenen Tabellen beim Importieren getriggert werden. Dies bedeutet, dass Datensätze mit Fremdschlüssel-Abhängigkeiten ohne zu kaskadieren eingespielt werden können. Die Fremdschlüsselabhängigkeiten bleiben erhalten. PostgreSQL ermöglicht diese komfortable Lösung mittels des Befehls „DISABLE TRIGGER“ und „ENABLE TRIGGER“, siehe Abbildung 4.

Die Entwickler checken jeden Tag ihren programmierten Quellcode auf dem Server ein. Die Testdatenbasen für die Entwicklungs- und die interne Testumgebung müssen daher täglich neu deployed werden, um die neuen Features anschließend testen und die Unittests ausführen zu können.

Für den Endanwendertest, welcher alle vier Wochen stattfindet, wird auch eine Build-XML-Datei mittels Ant-Targets in Eclipse in die dafür vorgesehene Umgebung deployed. Auch für die Schulungsumgebung wird eine Testdatenbasis mittels erstelltem Build mit dem Tool Jenkins eingespielt. Die Testdatenbasen für die Schulungsumgebungen werden automatisiert deployed (siehe Abbildung 3). Dies geschieht mit Hilfe einer entwickelten Batch-Datei, die die automatische Ausführung mit Hilfe der cmd.exe und in Verbindung des Windows-Tools „Aufgabenplanung“ anstößt.

Zusammenfassung

Die Qualität des gesamten Softwaretests steigt aufgrund der höheren Qualität der Testdaten. Die Entwicklung kann sich auf die eigentliche Arbeit konzentrieren, während das Testdatenmanagement sich ausschließlich um die Daten kümmert. Mit ihrem maximalen Nutzungspotenzial sind die Testdaten immer passend und aktuell für den täglichen Einsatz verfügbar. Bessere Testdaten sorgen für höhere Anwendungsqualität in allen Teststufen. Testdatenmanagement kann mit professionellen und an das Projekt angepassten Aufgaben und Maßnahmen stark zum Erfolg des Projektes beitragen. ■

Tools

- **Eclipse** ist eine moderne und mächtige Entwicklungssoftware.
- **Jenkins** ist ein erweiterbares, webbasiertes Softwaresystem zur kontinuierlichen Integration von Komponenten.
- **Jira** ist ein vielseitiges, webbasiertes Instrument für ein digitales Workflow-Management.
- **PostgreSQL** ist ein objektrelationales Datenbankmanagementsystem.
- Die **Windows-Aufgabenplanung** (Microsoft Management Console, MMC) führt Aufgaben automatisch aus.