Barbara Göller
(barbara.goeller@aitgmbh.de)
arbeitet als Consultant für Application Lifecycle Management (ALM) bei der AIT GmbH & Co KG. Ihre Schwerpunkte liegen auf Requirements- und Testmanagement, Anpassungen und Erweiterungen sowie dem Microsoft Visual Studio Team Foundation Server (TFS). Als ständige Autorin des TFS-Blogs (http://www.tfsblog.de) und Sprecherin im Microsoft-ALM-Umfeld gibt sie ihre Erfahrungen weiter.

Alexander Delp
(alexander.delp@aitgmbh.de)
arbeitet als Senior Consultant für Application Lifecycle Management bei der AIT GmbH & Co KG. Seine Schwerpunkte liegen auf Testmanagement und Automation, Continuous Integration sowie dem Begleiten von agilen und nicht-agilen Prozessen.

Keine Daten? Keine Tests! Effizientes Testdatenmanagement

„Managing test data nets real business value" [IBM]. Mit dieser Aussage räumt IBM einem Thema einen hohen Stellenwert ein, das in anderen Firmen häufig noch sehr stiefmütterlich behandelt wird: dem Testdatenmanagement. Wieso das Testdatenmanagement eine Grundvoraussetzung effizienten Testens ist, wie es bewältigt werden kann und warum dabei auch der Kunde mit ins Boot geholt werden sollte, beschreibt dieser Artikel.

Einer der Eckpfeiler hoher Softwarequalität ist das Testen. Doch wie wird die Qualität von Tests sichergestellt? Es ist kein Geheimnis, dass für hochwertige Tests die Testspezifikation, also die Beschreibung der Testfälle, als Grundlage für die Testausführung eine maßgebliche Rolle spielt. Zur Testspezifikation gehört jedoch nicht nur die Beschreibung der Testschritte. Ein klar definierter Ausgangszustand des zu testenden Systems ist ebenfalls unerlässlich. Denn ohne einen definierten Ausgangszustand kann auch das Ergebnis eines Testfalls nicht effizient beurteilt werden. Testfälle können nur in Kombination mit den richtigen Testdaten realitätsnah und damit qualitativ hochwertig beschrieben und ausgeführt werden.

Was die richtigen Testdaten sind und woher sie kommen? Die Beantwortung dieser Frage steht im Zentrum des Testdatenmanagements!

In diesem Artikel wollen wir für das häufig unterschätzte Thema des Testdatenmanagements sensibilisieren. Im ersten Teil stellen wir eine kurze Definition sowie die Relevanz hochwertiger Testdaten in den Vordergrund. Des Weiteren geben wir einen kurzen Überblick über Möglichkeiten und Fallstricke bei der Testdatenerzeugung. Im zweiten Teil stellen wir einen Prozess vor, welcher sich die Expertise aller beteiligten Rollen zunutze macht und unterschiedliche Ansätze kombiniert, um die Herausforderung des effizienten Testdatenmanagements zu meistern.

Einführung in das Testdatenmanagement

Was sind Testdaten und wieso muss man sie managen?

Testdaten umfassen alle Daten, welche vor der Ausführung eines Testfalls zur Verfügung stehen müssen [Spi12]. Sie setzen sich aus den Daten zusammen, die als Vorausset-

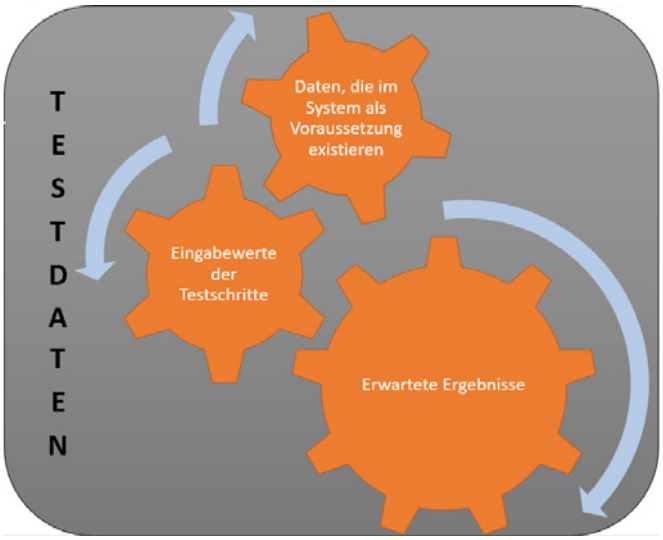

Abb. 1: Definition der Testdaten.

Statische Tabelle		
Account name	**Password**	**Expected login result**
imueller	****	Successful
arichter	****	Successful
imueller		Failed
rmeier	****	Failed

Testdatenpool
German street name
Auf der Au
Günther-von-Ebertshausen Allee
Hinter den Türmen
Pestalozzistraße
...

Abb. 2: Möglichkeiten zum Definieren von Testdaten für datengetriebene Tests.

zung im System existieren müssen, den Eingabewerten der Testschritte sowie den erwarteten Ergebnissen (vgl. Abbildung 1). Die Vollständigkeit und Verfügbarkeit der Testdaten ist essenziell für ein effizientes Testen: Klaffen Löcher in der Testdatenspezifikation, so müssen diese durch implizites Wissen während der Testausführung gestopft werden. Existieren keine passenden Daten im System, kann der Testfall gar nicht ausgeführt werden.

Eine Kernanforderung an Testdaten ist, dass sie möglichst realitätsnah sind. Wie die Infrastruktur, auf der ein Testsystem läuft, müssen auch die verwendeten Daten dem Produktivsystem so ähnlich wie möglich sein, um die Aussagekraft der Testaktivitäten zu maximieren. So wird beispielsweise die Anzeige eines Straßennamens besonders interessant, wenn Umlaute oder Sonderzeichen enthalten sind.

Die Aufgabe des Testdatenmanagements ist es sicherzustellen, dass alle benötigten Testdaten bei jeder Ausführung eines Testfalls zur Verfügung stehen. Der Arbeitskreis Software-Qualität und -Fortbildung e.V. (ASQF) [ASQF-a] benennt in seiner Prozessbeschreibung die dafür notwendigen Tätigkeiten Planung, Steuerung, Spezifikation, Konzeption, Bereitstellung, Berichterstattung sowie Archivierung [ASQF-b].

Die technisch anspruchsvollste dieser Aufgaben ist das Bereitstellen von im System vorausgesetzten Daten, was ein Beispiel verdeutlicht: Die Daten eines Beispielsystems verteilen sich über vier Datenbanken unterschiedlicher Technologien à 40 Tabellen. Eine dieser Datenbanken existiert aus lizenzrechtlichen Gründen nur als Produktivinstanz. Die Stammdaten eines Kunden sind in sieben Tabellen über zwei technologisch verschiedene Datenbanken verteilt. Um einen Kunden anzulegen, muss er mit einer Firma verknüpft werden. Zusätzlich müssen für den Kunden umfangreiche Profile sowie Vertragsbedingungen hinterlegt werden. Um einen neuen Kunden anzulegen, müssen somit über sieben Tabellen mit mindestens zwei unterschiedlichen Datenbanktechnologien befüllt werden.

Im Vergleich zu den komplexen technischen Rahmenbedingungen stellen Testfälle oft sehr überschaubare fachliche Anforderungen an die vorausgesetzten Daten. Statt einer vollständigen Spezifikation über 10 Tabellen werden eher einzelne Charakteristiken [Pric16] gefordert, welche für Testfälle tatsächlich relevant sind, wie „Ein Kunde mit ausstehenden Forderungen in Höhe von 5000€."

Eine solche Charakteristik kann im Testdatenmanagement zwischen zwei Extremen behandelt werden: „Versuche einen Kunden zu finden, welcher diese ausstehenden Forderungen hat." und „Es ist genau der zu verwendende Datensatz spezifiziert." Während die erste Vorgehensweise leicht zu konzeptionieren ist, birgt sie während jeder Testausführung einen hohen Zusatzaufwand und die Gefahr, gar keinen passenden Datensatz im System zu finden. Die zweite Vorgehensweise ist komplex in der initialen Konzeption, ermöglicht jedoch jederzeit eine problemlose, reproduzierbare Testausführung.

Spätestens mit Blick auf die Testautomatisierung ist es unumgänglich, ein solides Testdatenmanagement zu betreiben: Zum einen kann ein automatisierter Test keine Löcher durch implizites Wissen stopfen. Zum anderen ermöglicht ein solides Testdatenmanagement, datengetriebene Tests effizient einzusetzen. Bei einem datengetriebenen Test wird derselbe automatisierte Testfall mit unterschiedlichen Testdaten ausgeführt. Sind die Testdaten korrekt gewählt, erhöht sich dadurch die Testabdeckung, ohne weitere Testfälle automatisieren zu müssen.

Die Testdaten für einen datengetriebenen Test werden über eine statische Tabelle (vgl. Abbildung 2, links) oder über Testdatenpools (vgl. Abbildung 2, rechts) definiert. Eine statische Tabelle gehört zu genau einem Testfall und enthält pro Zeile alle Testdaten, welche dieser Testfall benötigt. Während der Ausführung wird der Testfall für jede Zeile einmal ausgeführt. Der zur statischen Tabelle in Abbildung 2 gehörende Testfall überprüft, ob ein Login mit den angegebenen Benutzerdaten das erwartete Ergebnis liefert. Ein Testdatenpool enthält eine Menge an möglichen Testdaten, aus denen während der Ausführung pro Iteration ein Element ausgewählt wird. Der Testdatenpool in Abbildung 2 kann von allen Testfällen verwendet werden, welche einen deutschen Straßennamen benötigen.

Strategien zur Testdatenerzeugung

Mit Fokus auf den Tätigkeiten Spezifikation, Erstellung und Bereitstellung von Testdaten sind die grundlegenden Strategien, Produktivdaten direkt beziehungsweise als Grundlage zu verwenden oder aber Daten in ihrer Gesamtheit zu synthetisieren. Der folgende Überblick beschreibt die Unterschiede und erläutert deren Vor- und Nachteile.

Der Hauptvorteil von Produktivdaten ist, dass mit geringem Aufwand reale Testdaten zur Verfügung stehen. Dem steht als Hauptproblem gegenüber, passende Datensätze zu finden, welche den von Testfällen geforderten Charakteristiken genügen. Die offensichtliche Voraussetzung für die Verwendung von Produktivdaten ist, dass bereits eine Produktivversion des Systems existieren muss. Handelt es sich um eine Neuentwicklung, so kann höchstens auf Produktivdaten ähnlicher Systeme zurückgegriffen werden.

```
// (1) als explizite Ausprägungen
TestdataProvider.NewUser
                .WithFirstName("Ida")
                .WithLastName("Müller");

// (2) dynamisch über Algorithmen
TestdataProvider.NewUser
                .HavingNameWithTypicalCharactersOf(Region.Germany)
                .HavingNameWith(Length.Medium);

// (3) dynamisch über Testdatenpools
TestdataProvider.NewUser
                .SelectNameFrom(Region.Germany);
```

Abb. 3: C#-Quellcode der Spezifikation unterschiedlicher Arten von Testdatencharakteristiken.

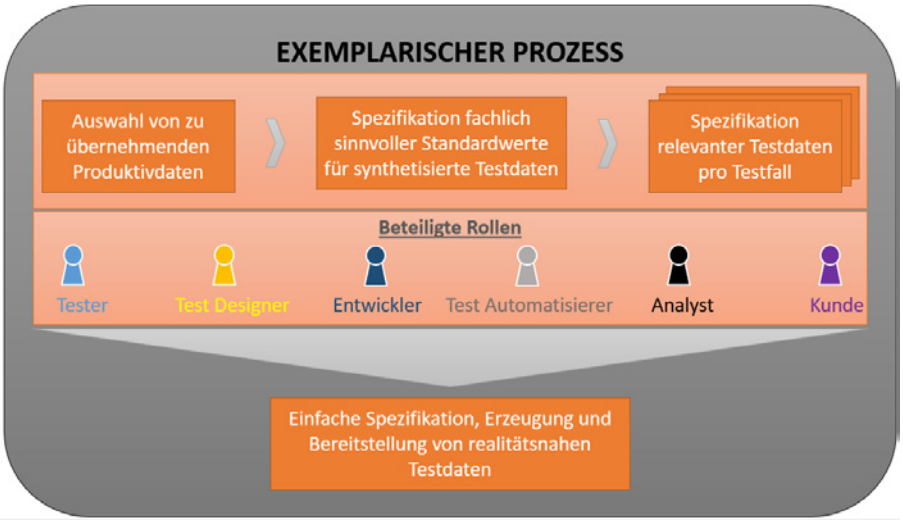

Abb. 4: Veranschaulichung des exemplarischen Testdatenmanagementprozesses.

Zu den Faktoren, welche den Aufwand für die Verwendung von Produktivdaten ansteigen lassen, zählt deren Alterung, da enthaltene Datumsangaben mit jedem Tag weiter in der Vergangenheit liegen. Des Weiteren müssen Produktivdaten, die beispielsweise personenbezogene Daten enthalten, vor der Verwendung nach datenschutzrechtlichen Aspekten untersucht werden.

Um Produktivdaten attraktiver zu machen, können sie vor ihrer Verwendung aufbereitet werden: Maskierung, Pseudonymisierung, die Änderung von Datumsangaben sowie das Anpassen einzelner Datensätze an durch Testfälle geforderte Charakteristiken helfen, den zuvor beschriebenen Problemen entgegenzuwirken. Dies resultiert jedoch in erheblichen Zusatzaufwänden und es muss genau darauf geachtet werden, dass durch Änderungen an Testdaten für einen Testfall nicht andere Testfälle ungültig werden.

Synthetisierte Daten werden künstlich – zumeist auf Basis von Systemspezifikationen oder Modellen – erzeugt. Die Möglichkeiten zur Erzeugung solcher Datensätze reichen von zufälliger Generierung über Verwendung fachlicher Algorithmen bis hin zu vollständig manueller Spezifikation. Synthetisierte Daten sind datenschutzrechtlich unbedenklich. Auch den Problemen der Alterung und des Fehlens passender Datensätze zu geforderten Charakteristiken kann durch eine bedarfskonforme Synthetisierung begegnet werden. Anders als bei der Verwendung von Produktivdaten stellt das Erhalten von realitätsnahen Testdaten aus Synthetisierung eine große Herausforderung dar und birgt somit die Gefahr von hohen Aufwänden oder mangelhaften Testdaten.

Das Beste aus zwei Welten

Im zweiten Teil wird ein möglicher Prozess vorgestellt, um der schwierigen Aufgabe effizienten Testdatenmanagements Herr zu werden. Getreu dem Motto: „Die Mischung macht's", folgt der implementierte Prozess dem Gedanken, dass in einem realen Projekt nicht ein Ansatz alleine die beste Lösung darstellen muss. Stattdessen basiert die Implementierung auf einer Kombination aus Produktiv- und synthetisierten Daten.

Im Fokus des Prozesses steht das Ziel, die Spezifikation, Erstellung und Bereitstellung von Testdaten so einfach wie möglich zu gestalten. Dadurch wird es ermöglicht, in einem Testfall die benötigten Testdaten nur anhand der für den Testfall relevanten Charakteristiken zu spezifizieren. Alle weiteren Daten, welche für die Testausführung in keiner spezifischen Ausprägung benötigt werden, werden im Hintergrund unter Zuhilfenahme von fachlich sinnvollen Standardwerten automatisch gesetzt. Die Art der Spezifikation ist dabei so gewählt, dass alle Beteiligten eine einheitliche, fachliche Sprache sprechen.

Wie bereits beschrieben liegt dem Prozess eine Mischung aus synthetisierten und Produktivdaten zugrunde. Dies bedeutet konkret, dass bereits relevante Produktivdaten aus bestehenden Systemen existieren und diese Produktivdaten unter Beachtung datenschutzrechtlicher Aspekte einmalig in den Testdatenbestand übernommen werden. Dieses Vorgehen eignet sich für Daten, welche nicht durch Testfälle mit konkreten Charakteristiken spezifiziert werden („Der Kunde kauft *ein Produkt*.") oder welche eine abgeschlossene Menge bilden („Wir liefern nach *Deutschland und Österreich*."). Ein Teil der Testdaten ist somit nicht nur möglichst realitätsnah, sondern spiegelt die Realität exakt wieder.

Wesentlich aufwendiger gestaltet sich die Erzeugung der Testdaten, die für Testfälle durch konkrete Charakteristiken gefordert werden. Deren Spezifikation findet in zwei Stufen statt. Zusammen mit Kunden und Analysten werden exemplarische, realitätsnahe Datensätze spezifiziert. Die besten Ergebnisse werden erzielt, wenn der Kunde dafür intensiv mit dem System interagiert. Eine reine Befragung birgt die Gefahr, dass die spezifizierten Daten ohne das Verständnis von Zusammenhängen wieder nur theoretische Qualitäten besitzen. Stattdessen sollte der Kunde die Möglichkeit haben, anhand eines Prototyps oder einer Vorabversion in Ruhe seine Geschäftsprozesse auszuführen. Die so spezifizierten Datensätze dienen im weiteren Verlauf als Standardwerte für die Testdatenerstellung.

Im zweiten Schritt werden für jeden Testfall die in den Testdaten geforderten Charakteristiken spezifiziert. Bei jeder Testdatenerstellung überschreiben diese Charakteristiken die zuvor festgelegten Standardwerte. Da Testfälle durchaus auch Randbedingungen überprüfen, müssen die Charakteristiken nicht mehr realitätsnah sein. Sie integrieren sich jedoch in einen ansonsten realitätsnahen Datenbestand.

Ein positiver Nebeneffekt des Vorgehens ist die Dokumentation der für einen Testfall relevanten Testdatencharakteristiken. Während sich Datensätze in der Datenbank gegebenenfalls über viele Tabellen erstrecken, ist durch die Form der Spezifikation direkt erkennbar, welche konkreten Werte für einen bestimmten Testfall relevant sind.

Was noch fehlt, ist, die Testdatencharakteristiken in einer einheitlichen, fachlichen Sprache spezifizieren zu können. Dafür müssen die Testdaten auf das Wesentliche reduziert werden. Die Menge der verbleibenden, von einem Testfall modifizierbaren Daten wird von ihrer Abbildung in der Datenbank abstrahiert. Das heißt, anstatt einer Eins-zu-eins-Abbildung der Datenbanken werden die verbleibenden Daten in fachliche Entitäten strukturiert und bereitgestellt.

In der Praxis ist es von Vorteil, Testdatencharakteristiken zunächst als explizite Ausprägungen zu betrachten (vgl. Verwendungsbeispiel in Abbildung 3 (1)). Dadurch wird die technisch herausfordernde Umsetzung der Abbildung von Datenbanken auf fachliche Entitäten von der Entwicklung einer dynamischen Beschreibung der Testdatencharakteristiken entkoppelt. Sind die technischen Voraussetzungen für das

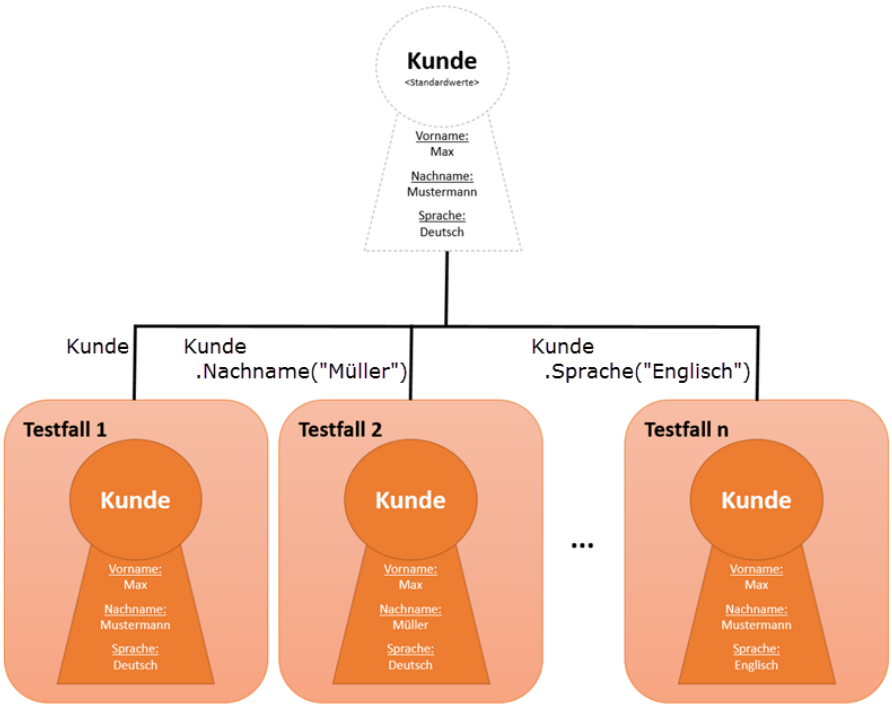

Abb. 5: Kunden-Beispiel.

Schreiben aller relevanten Daten in die Datenbanken erfüllt, kann die fachliche Sprache bei Bedarf um dynamische Testdatencharakteristiken erweitert werden. Der technische Aufwand beschränkt sich dann auf das Erzeugen konkreter Ausprägungen aus den Charakteristiken. Die Erzeugung der Daten erfolgt über Algorithmen (vgl. Verwendungsbeispiel in Abbildung 3 (2)) oder über Verwendung der bereits zuvor erwähnten Testdatenpools (vgl. Verwendungsbeispiel in Abbildung 3 (3)).

Zusammengefasst ergibt sich also folgender Prozess: Zunächst wird bestimmt, welcher Teil der benötigten Daten aus Produktivdaten übernommen werden kann und welcher Teil synthetisch erzeugt werden muss. Für die synthetisch zu erzeugenden Testdaten werden dann fachlich sinnvolle Standardwerte spezifiziert und hinterlegt. Fachliche Entitäten werden dem Verwender zur Erzeugung der Testdaten zur Verfügung gestellt. Zuletzt werden pro Testfall relevante Charakteristiken spezifiziert, welche die Standardwerte überschreiben. Der so definierte Prozess ist vereinfacht in Abbildung 4 dargestellt.

Es werde Licht

Ein Beispiel bringt Licht in das Dunkel der Theorie: Bei dem zu testenden System handelt es sich um die Neuentwicklung eines Onlineshops, welche den Vorgänger ablösen soll. Damit ein Kunde Einkäufe tätigen kann, muss ein Account für ihn angelegt werden. Für die angebotenen Produkte werden deren Daten im System benötigt.

Da sich die angebotene Produktpalette des Onlineshops nicht verändert, können beispielsweise die Daten der angebotenen Produkte aus dem Altsystem übernommen werden. Sie stellen alle relevanten Ausprägungen in diesem Bereich dar und müssen somit nicht für einen Testfall modifizierbar sein.

Daten, welche im neuen Onlineshop erhoben beziehungsweise verarbeitet werden, werden hingegen nicht aus den Produktivdaten des Altsystems kopiert. Sie werden im ersten Schritt der Synthetisierung mit realitätsnahen Standardwerten spezifiziert. Für jeden Testfall, welcher zwar einen Kunden mit gültigem Account erfordert, für den der Name jedoch nicht von Relevanz ist, wird somit beispielsweise ein Kunde mit dem Namen Max Mustermann angelegt (vgl. Abbildung 5, Testfall 1).

Im zweiten Schritt der Synthetisierung werden alle für einen Testfall relevanten Daten spezifiziert. Ist nun also der Name eines Kunden für einen Testfall relevant, um beispielsweise die Anzeige von Sonderzeichen zu prüfen, so spezifiziert der Testfall diesen Namen explizit und überschreibt damit den hinterlegten Standardwert (vgl. Abbildung 5, Testfall 2 + Testfall n). Wie eingangs behauptet, erlaubt das beschriebene Vorgehen, nur relevante Testdatencharakteristiken pro Testfall spezifizieren zu müssen, während alle anderen notwendigen Daten im Hintergrund automatisch mit fachlich sinnvollen Standardwerten aufgefüllt werden.

Um das gegebene Beispiel mit Bezug auf die fachliche Spezifikation fortzuführen: Die Daten eines Kunden befinden sich in der zugrunde liegenden Datenbank in zwei verschiedenen Tabellen (Adressdaten und Login-Daten). Jede der Tabellen enthält Zeitstempel für die Erzeugung und die letzte Änderung einer Zeile. Gegen diese Daten sind zunächst keine Testfälle vorgesehen, weshalb die Datumsstempel nicht Teil der zur Verfügung stehenden fachlichen Entitäten sind. Trotz der Aufteilung der Daten eines Kunden auf mehrere Tabellen wird der Kunde als eine fachliche Entität behandelt und der Testdatenspezifikation als solche zur Verfügung gestellt.

Was es auf dem Weg noch zu gewinnen gibt

Die technische Umsetzung der beschriebenen Testdatenerzeugung bedingt eine intensive Auseinandersetzung mit dem fachlichen Domänenmodell, der Analyse und dem technischen Datenschema der Entwicklung. Befindet sich das zu testende System in der Neuentwicklung, hat dieses intensive Auseinandersetzen einen Review-Charakter: In enger Zusammenarbeit mit Analysten und Entwicklern wird die Umsetzung des Domänenmodells aus Sicht des Tests beleuchtet und gegebenenfalls überarbeitet.

```
// Arrange
TestdataProvider.NewUser
    .WithAccountName("imueller")
    .WithPassword("****")
    .WithFirstName("Ida")
    .WithLastName("Müller")
    .CreateInDatabase();

// Act
var onlineShopClient = new OnlineShopClient();
onlineShopClient.Login("imueller", "****");

// Assert
onlineShopClient.ShouldShowDisplayName("Ida Müller");
```

Abb. 6: C#-Quellcode eines beispielhaften Testfalls inkl. Testdatengenerierung.

Die Implementierung der vorgestellten Testdatenerzeugung wird den Verwendern für Tests in Form einer Programmierschnittstelle zur Verfügung gestellt. Dieses API gliedert sich technologisch in die Entwicklung und in die Testautomatisierung. Durch die Verwendung derselben Programmiersprache wird eine durchgängige Programmierung der Testdatenerzeugung und des Testablaufs ermöglicht (vgl. Abbildung 6).

Die Implementierung der Testdatenerzeugung gehört neben die Produktionsquellen in die Quellcodeverwaltung. Damit ist eine zu den jeweiligen Produktionsständen passende Versionierung sichergestellt. Gleichzeitig wird die Archivierung von Testdatenbanken hinfällig, da alle Testdaten jederzeit reproduziert werden können.

Fazit

Erst durch die richtigen Testdaten können Testfälle realitätsnah und qualitativ hochwertig ausgeführt werden. Die Relevanz eines adäquaten Testdatenmanagements wurde im ersten Teil dieses Artikels hervorgehoben. Darauf aufbauend wurde im weiteren Verlauf ein möglicher Prozess vorgestellt, welcher auf der kombinierten Verwendung von Produktiv- und synthetisierten Daten basiert, um ein effizientes Testdatenmanagement zu ermöglichen.

Der initiale Aufwand des beschriebenen Prozesses ist hoch, jedoch gibt es Testdatenmanagement niemals geschenkt! Der Aufwand muss initial investiert werden, da sonst im Nachgang bei jeder Testfallausführung Probleme aufkommen können.

Dem hohen initialen Aufwand steht der anschließende, kontinuierliche Nutzen gegenüber: Verständliche, bedarfsgerechte Spezifikation sowie reproduzierbare Erstellung und Bereitstellung hochwertiger Testdaten. ■

Entwicklungshistorie

Initial wurde der in diesem Artikel beschriebene Ansatz für automatisierte Oberflächentests einer bestehenden Client/Server-Anwendung entwickelt, welche Dokumente mit Metadaten verknüpft. Die Daten verteilen sich auf etwa 20 Tabellen in einer SQL-Datenbank auf dem Server. Die Testfälle werden von Fachentwicklern automatisiert, für die ein sprechendes und schlankes API zum Erzeugen von Testdaten bereitgestellt werden sollte.

Aufgrund der positiven Ergebnisse bezüglich der einfachen Handhabung und der guten Lesbarkeit wurde der Ansatz für den Folgerelease erweitert: Zum einen steht das API zum Erzeugen der Testdaten nun in einer zweiten Ausprägung bereit. Diese zweite Ausprägung ist umfangreicher und wird von den technischen Entwicklern beim Automatisieren von Integrations- beziehungsweise Systemtests verwendet. Zum anderen wurden die vorhandenen internen Datenstrukturen verwendet, um neben der Erzeugung von Testdaten *vor* einer Testausführung auch das Auslesen der Ergebnisdaten *nach* einer Testausführung aus den Datenbanken zu erlauben. Dadurch werden die durch einen Testfall veränderten Daten nicht mehr mittels dem zu testenden System, sondern direkt in den Datenbanken geprüft.

Die aktuelle Ausprägung des beschriebenen Ansatzes wird für die Testautomatisierung einer Webanwendung zum Erstellen und Verwalten von Leasinganfragen umgesetzt. Die Daten verteilen sich hauptsächlich auf etwa 60 bis 80 Tabellen in fünf SQL-Datenbanken. Da es sich bei dem Produkt um eine aktuelle Neuentwicklung handelt, fallen am Datenmodell sowie dem Datenbankschema stetig Änderungen an. Die Entwicklung des in diesem Artikel beschriebenen Ansatzes wird daher als Chance genutzt, um eine Instanz für zeitnahe Reviews solcher Änderungen zu haben.

Literatur & Links

[ASQF-a] Arbeitskreis Software-Qualität und -Fortbildung e.V., siehe: https://www.asqf.de/

[ASQF-b] ASQF, Testdatenmanagement – eine Prozessbeschreibung in sieben Schritten, SQ-Magazin, Juni 2014, siehe: https://www.asqf.de/tl_files/asqf/bilder/publikationen/140601%20ASQF%20Testdatenmanagement%20Prozess.pdf

[IBM] Back to basics – Fundamentals of test data management, IBM software, 2012, siehe: http://www.informationweek.com/pdf_whitepapers/approved/1345732672_back_to_basics.pdf

[Pric16] Test Talks: Episode 94: H. Price: Agile Modeling & Test Data Management Archeology, 13.3.2016, siehe: https://joecolantonio.com/testtalks/94-huw-price-agile-modeling/

[Spi12] A. Spillner, T. Linz, Basiswissen Softwaretest, 5. Auflage, dpunkt.verlag, 2012, S. 263