Dietmar Landgraf / Tadeusz Skolka

Praxistaugliche **Test**methodik für agile Systementwicklung am Beispiel eines **Treasury Systems**

Standardsysteme im Bereich Treasury gibt es seit mehreren Jahren. Sie wurden in der Vergangenheit bei Banken entwickelt und anschließend – ergänzt um bei mehreren Kunden nachgefragte Funktionalitäten - am Markt angeboten. Sie verfügen über typische Berichte und Funktionen, die generell bei allen Banken verwendet werden.

Make or buy

Der Einsatz von Treasury Systemen bei mittelständischen Banken wird durch diverse Faktoren begrenzt, die dem Kauf der erforderlichen Software entgegenstehen. Es wird zunehmend beobachtet, dass sich mittelständische Banken gegen marktgängige und somit für eigene Lösungen entscheiden. Es gibt dafür mehrere Gründe, wovon die wichtigsten, nämlich Funktionsumfang und Kosten, hier angeführt werden.

Die Kosten eines Treasury Systems beziehungsweise eines Moduls sind zwar generell nicht gering (je nach Hersteller), stellen im Normalfall aber keine wesentliche Hürde bei einer Systementscheidung dar. Schwieriger wird es, wenn auch noch die Kosten der Systemimplementierung/-parametrisierung und der Integration mit einkalkuliert werden müssen, da sie in der Regel (je nach Komplexität der bereits bestehenden Infrastruktur in der Bank) hoch und mit Unsicherheiten belastet sein können. Hinzu kommen Kosten der Schulung und der Einarbeitung in das neue System sowie Kosten für das Helpdesk. Zudem periodische, gelegentlich relativ hohe Kosten für die Installation sowie für Tests neuer Releases. Diese dürfen gemäß Update-Strategie des Softwareherstellers für gewöhnlich vom Finanzinstitut nicht abgelehnt werden, da ansonsten das System durch den Hersteller nur für begrenzte Zeit noch unterstützt werden kann. Firmeninterne Schätzungen bei Trisolutions GmbH teilen die Kosten in Treasury-Implementierungsprojekten folgendermaßen auf: 5 bis 10 Prozent Lizenzkosten, 10 bis 20 Prozent Parametrisierung des Systems, 70 bis 80 Prozent Schnittstellen/Datenaufbereitung.

Neben den erwähnten allgemeinen Kosten ist die benötigte Funktionalität der Auslöser für die Entstehung weiterer Kostenblöcke. Zur Vereinfachung werden hier die drei wesentlichen Punkte hinsichtlich des Funktionsumfangs dargestellt, die bei der Entscheidung „make or buy" eine besondere Rolle spielen.

– Es gibt in der Regel nur wenige Spezialisten für die Systempflege. Die Honorare der guten Systemspezialisten für marktgängige Lösungen sind teilweise hoch, da sie bei vielen verschiedenen Finanzinstituten für Anpassung beziehungsweise Weiterentwicklung der Systemkomponenten eingesetzt werden. Dagegen wird das Systemknowhow der Spezialisten einer eigens entwickelten Lösung vom Markt nicht nachgefragt.

– Es wird eine einfache, aber auf die gegebenen Prozesse zugeschnittene bankspezifische Funktionalität gebraucht. Der Aufwand der Anpassung eines auf dem Markt erworbenen Systems kann daher größer sein als die eigenständige Programmierung der geforderten Funktionalität.

– Es wird beim Kauf eines marktgängigen Systems grundsätzlich eine umfangreiche Installation und Parametrisierung benötigt. Der Grund ist einfach, denn ohne diese Parametrisierung können sogar die einfachsten Funktionen generell nicht aktiviert werden. Dies bedeutet, dass unter Umständen auch die Erstellung eines einfachen Berichts, den ein IT-Spezialist mit wenigen SQL-Abfragen aus einem Data Warehouse generieren kann, innerhalb einer marktgängigen Lösung erst nach einer umfangreichen Parametrisierung und gegebenenfalls einer Schnittstellenerstellung zur Verfügung gestellt wird.

Technik und Tests

Nach Abwägung verschiedener Kriterien, wovon hier nur die wichtigsten vorgestellt wurden, hat sich das als Beispiel gewollte Finanzinstitut für die Erstellung einer Eigenlösung für das Aktiv-Passiv-Management Reporting (APM) entschieden.

Technischer Überblick: Aus vier Vorsystemen werden Daten zu Kapitalmarkt-, Kredit- und Sparprodukten über drei Transformationsebenen in ein Data Warehouse überführt. Die Database „Detail" enthält Rohdaten im gleichen Format wie im Vorsystem abgelegt. Der „Business Layer" vereinheitlicht die verschiedenen Formate und speichert neu generierte Daten für das Reporting (zum Beispiel CF-Komponenten beziehungsweise CF-Shifts). Die für das APM-Reporting relevanten Daten werden anschließend in den sogenannten

Dietmar Landgraf, Risikocontrolling, IKB Deutsche Industriebank AG, Düsseldorf, und Tadeusz Skolka, TriSolutions GmbH, Hamburg

Am Beispiel der Einführung eines neuen Systems für das Aktiv-Passiv-Management-Reporting beschreiben die Autoren die wichtigsten Überlegungen bei der Umsetzung in die Praxis. Zunächst ging es in der Bank um die Frage, ob es für das Haus sinnvoller ist, ein Standardsystem anzupassen oder eine eigene Anwendung zu programmieren. Nach der Entscheidung für eine eigene Lösung und die anzuwendende Entwicklungsmethode, standen umfangreiche Überlegungen zur Auswahl der Testumgebung, deren Einsatz und das Reporting über die Tests an. (Red.)

„Rechenkern" überführt. Dabei werden fehlerhafte Produktabbildungen und Cashflows aus Vorsystemen korrigiert und die Daten soweit aufbereitet und strukturiert, dass über ein separates APM-Berichtstool die gewünschten Berichte erzeugt und angezeigt werden können (siehe Abbildung 1).

Tests in der agilen Umgebung: Die Entscheidung für eine agile Entwicklungsmethode, die durch ständige Änderungen an der Software (Refactoring) gekennzeichnet ist, hat direkte Auswirkungen auf die Tests. Insbesondere nimmt die Testhäufigkeit signifikant zu. Außerdem kann sich die Durchführung der Tests wesentlich schwieriger und aufwendiger als beim Einsatz des bekannten, klassischen Wasserfallmodells[1)] gestalten[2)]. Vor diesem Hintergrund muss das Testmanagement zwangsläufig Schwerpunkte auf automatisierte Regressionstests setzen.

Hinzu kommt, dass das Fehlen der kompletten fachlichen Detailvorgaben zum Zeitpunkt der Testdurchführung in der agilen Umgebung eher die Regel als die Ausnahme ist, wodurch die generellen Testziele sowie die Ziele der einzelnen Tests unklar erscheinen können.[3)] Die Tatsache der häufigen Softwareanpassung beziehungsweise -weiterentwicklung ändert den Charakter der fachlichen Tests insoweit, als sie meist den Entwicklertests gleichgesetzt werden, bei der die Fehleridentifizierung noch im Aufgabenbereich des Programmierers liegt. Durch kontinuierliches Testen werden die Ad-hoc-Spezifikationen auf ihre Validität geprüft und gegebenenfalls angepasst.[4)]

Agile Ansätze

Die Umstellung der IT-Vorgehensweise von der Wasserfallmethode auf agile Ansätze erfordert eine Änderung der Projekt- und Denkkultur. Vor allem dann, wenn der Einsatz der agilen Methode im Finanzinstitut noch keine lange Tradition hat, können sich kritische Stimmen der Projektverantwortlichen wegen des „unangemessenen Aufwandes" bei der Testdurchführung mehren, die schließlich im Zuge außerordentlicher Teststraffungsversuche zum Verlust der Testkonsistenz und -qualität führen können. Dennoch kann hierüber deutlich Zeit gespart werden.

Im beschriebenen Projektbeispiel wurde nach der Übergabe des entwickelten Systems in die Produktion die bisherige informelle Art der Systemanpassung wesentlich gestrafft. Die Änderungen wurden wegen der erhöhten Anforderungen der Produktionsumgebung noch vor der Umsetzung je nach Priorität zu Releasen zusammengefasst. Die Tests sowie die Produktionsübernahmen wurden dementsprechend auch releaseorientiert durchgeführt.

Die Umgehung der organisatorischen Schwierigkeiten, die sich in agilen Projekten schnell verbreiten können, kann häufig durch eine gewisse Straffheit der Testmethodik gewährleistet werden. Sie beginnt mit der Transparenz hinsichtlich der Positionierung der Tests im Projekt inklusive möglicher Schnittstellen zu fachlichen Bereichen und zur DV-Umsetzung, der Testkonzeptionierung und endet mit der Dokumentationsmethodik der durchgeführten Tests. Die Hauptaufgabe des „großen Testwurfs" besteht darin, die Schnittstellen zwischen einzelnen Projektbereichen zu identifizieren und den Informationsaustausch zwischen Entwicklung, Tests und fachlicher Analyse zu institutionalisieren. Dies soll Klarheit, Transparenz und Reibungslosigkeit der Testaktivitäten verstärkt unterstützen.

Testansatz: Im nachfolgenden Text werden die wesentlichen Eckpfeiler für den Aufbau der Tests in der agilen Umgebung beschrieben: Testarten, Testobjekte, Technische Durchführung, Testsystem, Testreporting.

In Abhängigkeit von der gewählten Entwicklungsmethode bestimmt sich, welche optimalen Testarten im Projekt verwendet werden sollten. Nach einer entsprechenden Bedarfsanalyse wurden in vorliegendem Beispiel drei Testarten (Smoke Tests, Regressionstests und Abnahmetests) ausgewählt.

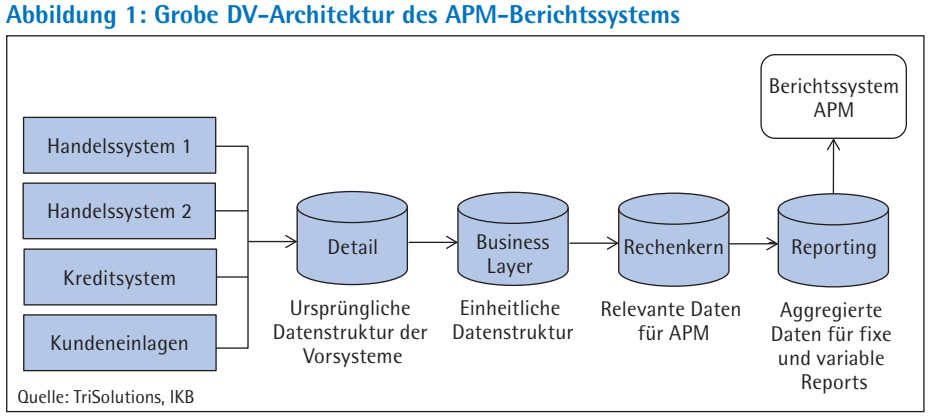
Einzelne Module beziehungsweise ausgewählte Hauptfunktionalitäten wurden zunächst im Rahmen sogenannter Smoke Tests auf grobe Plausibilität überprüft, die meist sofortige DV-Maßnahmen nach sich zogen. Bei Smoke Tests handelt es sich um Probeläufe eines Systems, die Probleme offenlegen sollen, welche ernst genug sind, um das Programm nochmals zu überarbeiten. Parallel dazu wurden Regressionstests zur Plausibilisierung der Datenwelt konzipiert und automatisiert. Nach Abschluss der Smoke Tests stand eine erste Version der „testbaren" Software für Akzeptanztests zur Verfügung.

Regressionstests

Die Regressionstests haben bei agilen Projekten eine besondere Bedeutung. Sie sollten häufig, bestenfalls täglich durchgeführt werden. Dadurch kann sichergestellt werden, dass bereits erfolgreich getestete Funktionalitäten auch nach DV-Änderungen in Softwarekomponenten weiterhin korrekt funktionieren. Regressionstests müssen sorgfältig ausgewählt werden – sinnvollerweise aus dem Pool bereits getesteter Einzeltestfälle. Zudem müssen fachliche und performancebezogene Gesichtspunkte bei der Implementierung der Einzeltestfälle in die Regressionstests beachtet werden. Typischerweise handelt es sich daher um relativ einfache Prüfungen, die den Gesamtbestand der Daten innerhalb einer Datenbasis betreffen.

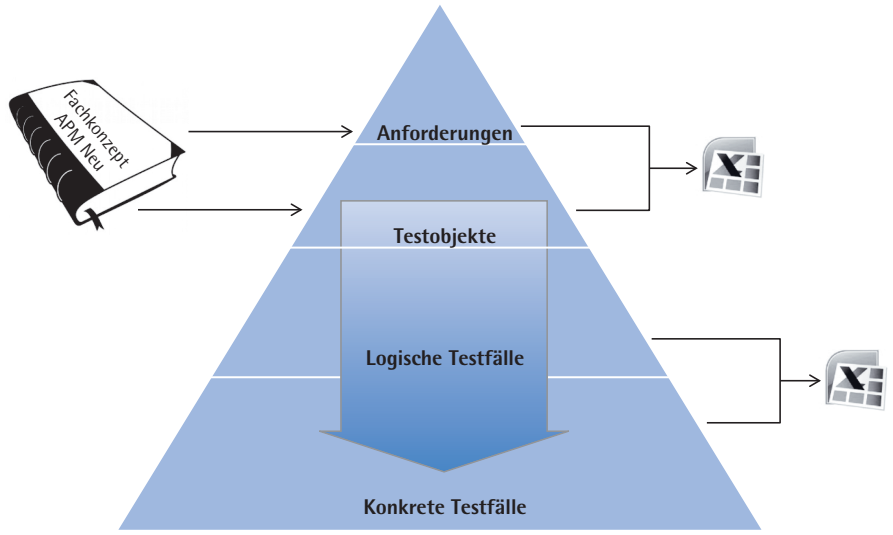
Die Performance ist eine der wichtigsten Anforderungen an Regressionstests. Mangelnde Performance, das heißt zu hohe Anforderung an die Rechenkapazität (und somit zu teuer) oder zu hoher Verwaltungsaufwand (häufige Testabbrüche) führen oft zu Widerständen gegen die Regressions-

Abbildung 1: Grobe DV-Architektur des APM-Berichtssystems



Quelle: TriSolutions, IKB

Abbildung 2: Ableitung der konkreten Testfälle



Quelle: TriSolutions, IKB

tests. Gut funktionierende Regressionstests entscheiden über die Daten- und Berichtsqualität. Fehlende beziehungsweise unzureichende Regressionstests sind häufig Ursache von fehlerhaften fachlichen Berichten (rubbish in, rubbish out).

Testobjekte identifizieren

Ein Testobjekt ist eine Prozedur, Funktion beziehungsweise Methode, deren Umsetzung im Rahmen der Tests erfolgt.[5] Deshalb besteht eine der wichtigsten Aufgaben darin, die relevanten Testobjekte zu identifizieren. Grundlage zur Identifizierung der Testobjekte ist im Normalfall ein Fachkonzept, das alle wesentlichen fachlichen Prozeduren, Funktionen und Methoden beschreibt. Die agilen Projekte bringen häufig zusätzlich mehrere verkürzte Fassungen fachlicher Vorgaben, die zwar noch nicht in das Fachkonzept übernommen wurden, deren technische Umsetzung aber bereits verfügbar ist und für Tests zur Verfügung steht. Sie sind dementsprechend ebenfalls für die Identifizierung der Testobjekte heranzuziehen.

Gelegentlich kommt es zudem vor, dass die Anforderungen, die zu Beginn des Projekts anhand des Fachkonzepts identifiziert wurden, in eine weitere fachlich detailliertere Vorgabe für die Umsetzung der Funktionalität in der Software fließen, zum Beispiel bei diversen fachlichen Berichten oder bei einer Erläuterung der Kalkulation bestimmter Kennzahlen. Diese sind dann dementsprechend auch zu berücksichtigen (siehe Abbildung 2).

Daher kann die Identifizierung der Testobjekte unter Umständen mehrere Bearbeitungsstufen bedeuten. Aus den Testobjekten werden logische Testfälle abgeleitet, die wiederum in ein oder mehrere, in Tests durchführbare, konkrete Testfälle münden. Das Endergebnis der Analyse mündet in einem Testobjektkatalog. Bei jedem Testobjekt gibt es Referenzen zum entsprechenden Abschnitt des Fachkonzepts. Dadurch kann sichergestellt werden, dass keine wesentlichen Inhalte des Fachkonzepts außer Acht gelassen werden.

Technische Durchführung

Bei Data Warehouse orientierten Projekten handelt es sich um die Verarbeitung von großen Datenvolumina. Im Bereich APM, wo für das Reporting beispielsweise verschiedene CF-Komponenten generiert werden, kann das Datenvolumen schlussendlich die zehnfache Menge (und mehr) der originären Bestände betragen. Aufgrund der Größe der Datenmenge kann die Performance infolge rechenintensiver Verarbeitung von Testabfragen einbrechen und die Tests negativ beeinflussen. Je nach Umfang der zu testenden Daten können verschiedene Plattformen verwendet werden: Excel, Access, ODBC, Oracle. Im vorliegenden konkreten Falle erfolgte die Entwicklung und Durchführung von Tests und Regressionstest zunächst mit Excel. Durch die IT-Abteilungen wurden je Vorsystem Datenabzüge passend für die zu testenden Sachverhalte bereitgestellt. Diese Vorgehensweise stellte sich allerdings recht bald aus nachfolgenden Gründen als nicht praktikabel heraus:

– Datenextrakte mit mehreren zehntausend Zeilen: Nach Ergänzung um Formeln dauerten Berechnungen bei einem einfachen Datenabgleich per SVERWEIS mehrere Minuten oder führten zum Programmabsturz. Einfache Plausibilisierungen ausgewiesener Beträge per Matrixfunktion oder SUMMEWENN-Funktion waren praktisch unmöglich.

– Sich änderndes Datenlayout: veränderte Reihenfolge von Datenspalten zum Beispiel durch neu eingefügte Datenspalten oder gelöschte Datenspalten.

In einem nächsten Entwicklungsschritt erfolgte das Testen mit Access-Datenbanken. Mit der IT wurde ein standardisiertes Datenlayout für die Anlieferungen vereinbart, sodass die angelieferten CSV-Dateien entweder in Access-Datenbanken verknüpft oder importiert werden konnten. Aufgrund des Datenvolumens wurde je Vorsystem eine Testdatenbank angelegt, in welche die Daten des DWH-DTL und DWH-BL importiert wurden. Anschließend wurden Abfragen erstellt, um die korrekte Überführung der Daten von dem DWH-DTL in den standardisierten Business Layer zu plausibilisieren. Der Nachteil dieser Herangehensweise:

– Vom vereinbarten Datenlayout abweichende Datenanlieferungen machten Datenbankabfragen teilweise unbrauchbar.

– Neu ergänzte Datenfelder wurden unter Umständen nicht immer kommuniziert und wurden entsprechend nicht importiert oder von den bestehenden Datenbankabfragen nicht berücksichtigt.

– Standardisierte Anlieferung enthielt je nach Vorsystem etliche redundante Datenfelder.

– Mit fortschreitender Entwicklung des Business Layers stieg auch das Volumen der angelieferten Daten.

Die Access-Datenbanken kamen sehr schnell an die Kapazitätsgrenze von 2 GB,

sodass ein alternativer Weg der Datenversorgung gewählt werden musste.

Durchführung mit Access (ODBC-Schnittstelle)

Um das Problem mit den großvolumigen CSV-Dateien zu lösen, hat die IT den Testern ermöglicht, per ODBC-Schnittstelle direkt auf die Oracle-Datenbanken zuzugreifen. Die zuvor als CSV-Datei angelieferten Daten wurden nun über ODBC-Zugriff auf eine Materialized View in Access abrufbar.

Die Vorteile lagen zunächst offensichtlich auf der Hand:

– Testdaten konnten direkt und ohne Zeitverlust an der Quelle abgerufen werden.

– Aufwendige Berechnungen konnten teilweise auf der Oracle-Datenbank durchgeführt werden, sodass die Testabfragen in den Access-Datenbanken schneller durchliefen.

Es gibt jedoch auch Nachteile:

– Datentypprobleme (Decimal versus Double) machten Type-Cast-Abfragen in Oracle erforderlich.

– Sich verändernde und vom vereinbarten Standard abweichende Materialized Views führten auch hier zu Problemen in den Testabfragen in Access.

– Die fortschreitende Entwicklung des Business Layers führte zu weiter steigendem Datenvolumen in den Materialized Views, welche die Access-Datenbanken bald wieder an die Kapazitätsgrenze von 2 GB führten.

Voranstehende Herangehensweisen haben gezeigt, dass ein sinnvolles Testen der Daten nur über den direkten Zugriff auf die Datenquelle (Oracle-Datenbank) möglich war. Daher wurde den Testern durch die IT ein direkter Zugriff auf einen eigens für die Projektentwicklung/-umsetzung eingerichteten Testuser (mit beschränkten Zugriffsrechten) der Datenbank eingerichtet.

Als Vorteile sind zu nennen:

– Abfragen können direkt zum Beispiel über den SQL Developer in Oracle ausgeführt werden.

– Alternativ können die Abfragen auch per VBA und ODBC-Schnittstelle beispielsweise durch ein Excel-Tool an die Datenbank übertragen und Ergebnisse abgerufen werden.

– Auswertung der Ergebnisse/Zwischenergebnisse aufgrund des beschränkten Datenvolumens in Excel flexibel möglich.

– Signifikanter Performancegewinn gegenüber den vorherigen Testherangehensweisen.

– Möglichkeit der Testautomatisierung und Testwiederholung (Regression) durch VBA.

Auch hier sind jedoch auch Nachteile in Betracht zu ziehen:

– Zusätzlicher Entwicklungsaufwand für das „Übersetzen" der bereits bestehenden Access-Abfragen in die Oracle-SQL-Syntax.

– Datenbankzugriff musste streng reguliert werden, da grundsätzlich ein direkter Zugriff auf Abnahme- und Produktionsdaten möglich war.

Es gibt marktgängige Testsysteme, die typische organisatorische Testaufgaben mit Erfolg unterstützen. Dazu gehören beispielsweise HP ALM und SAP Solutions Manager. Die Verwaltung der in Tests festgestellten Fehler und Retests werden in Ticketing-/Issue-Systemen wie zum Beispiel Jira und Mantis durchgeführt.

Im Projektbeispiel wurde für Testzwecke die Nutzung von Mantis vereinbart – das System hat zwar keine vollständige Testfunktionalität, erlaubt jedoch die automatische Übernahme der Testfälle aus Excel Sheets und die Verwaltung der identifizierten Fehler (inklusive Testdokumentation). Die Erfassung und Verwaltung der Testobjekte sowie der logischen Testfälle wurde dagegen in einem speziell entwickelten Excel Sheet durchgeführt.

Berichte

Bei erfolgreichen Tests wurden bisweilen bis zu drei Berichtsarten verwendet. Und zwar interner Testbericht, Testüberblick und detaillierter Testbericht.

Der interne Testbericht hat zum Ziel, den aktuellen Zustand der Tests abzubilden. Aufgegliedert nach Aufgabengruppen zeigt der Bericht die generellen Aktivitäten und insbesondere die wesentlichen Abhängigkeiten, die gegebenenfalls weitere Testdurchführungen verhindern. Es geht vor allem um eine Effizienzsteigerung und das Erkennen potenzieller Stolpersteine bei der Testvorbereitung und -durchführung. Wichtig dabei ist auch die Einfachheit der Berichtsausgestaltung, damit Tester und die Projektleitung in kürzester Zeit die Zusammenhänge erkennen können. Da einzelne Testwochen unterschiedliche Schwerpunkte haben können, ist eine tägliche Überprüfung der Testaktivitäten des internen Testberichts sehr hilfreich für die Testkoordination und -überwachung.

Im Gegensatz dazu wird für höhere Managementebenen (siehe Abbildung 3) eine eher generellere Sicht auf die Testblöcke mit dem Grad der Testerfüllung benötigt, damit auch diese sich in kurzer Zeit ein Bild von dem Testfortschritt machen können. Diverse Erläuterungen/Fußnoten haben eher einen abgrenzenden Charakter und offenbaren weitere Details, die im generellen Bild nicht direkt erkennbar sind.

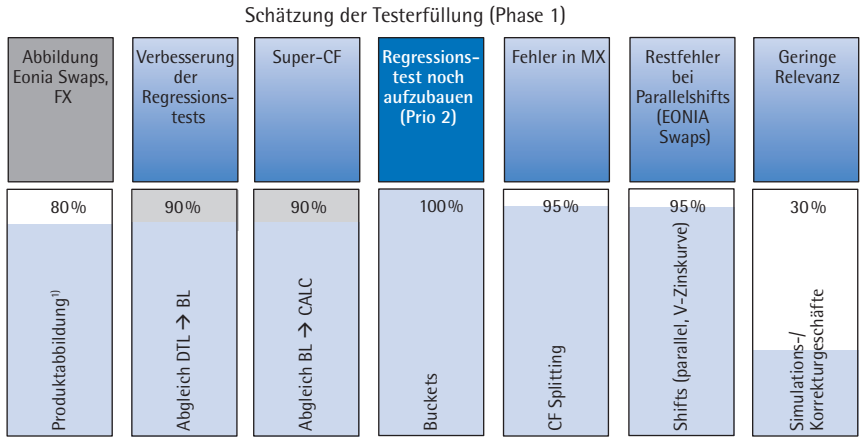
Eine letzte Reportingmöglichkeit ist ein detaillierter Testbericht in tabellarischer Form, der meist erst zum Schluss einer Phase beziehungsweise des Projektes erstellt wird. Seine Aufgabe ist es, Testressourcen und Managementaufmerksamkeit auf jene Aufgaben zu fokussieren, die den zu erzielenden Phasenschlusstermin potenziell gefährden könnten. Mithilfe farblicher Unterlegung kann visuell auf Gefahrenbereiche hingewiesen werden. Die Test- und Projektleitung hat dadurch die Möglichkeit, die Wirkung der bereits getroffenen Maßnahmen gegebenenfalls täglich zu beobachten und durch geeignete Maßnahmen (beispielsweise Verlagerung von Ressourcen) Problembereiche kurzfristig zu entschärfen.

Lessons Learned

Vor dem Hintergrund der Projekterfahrungen lassen sich einige kritische Erfolgsfaktoren für Tests in agilen Entwicklungsprojekten im Bereich Treasury formulieren:

– Die Entwicklung der Testvorgehensweise und -struktur sollte so früh wie möglich beginnen. Dadurch lassen sich die Anliegen des Testteams besser im Projekt, aber vor

Abbildung 3: Beispiel eines Status-quo-Berichts für das höhere Management



1) Produkte, deren CF an der Schnittstelle zu APM neu generiert werden: FX (Spot, Forward, Swap), Call Accounts, Eonia Swaps, FRAs, Zero-Bonds (OK), thesaurierende Produkte (OK), Tagesgeld (OK), Kontokorrentkonten (OK)
2) Es handelt sich um Testfälle mit kritischer bis großer Relevanz für T1-Planung. TF mit geringer Relevanz wurden in die Auswertung nicht mit einbezogen.

Quelle: TriSolutions, IKB

allem auch in der Entwicklung platzieren und umsetzen.

– Der Erfolg der Tests ist unmittelbar davon abhängig, wie sie fachlich strukturiert und mit dem Fortschritt in der Entwicklung synchronisiert wurden.

– Der Erfolg der Tests ist direkt davon abhängig wie eng die IT-Entwicklung und die Entwicklung der Tests abgestimmt sind. Wenn nicht „verstanden" wird, wie bestimmte Sachverhalte von der IT umgesetzt wurden, können Testherangehensweisen unter Umständen „in die falsche Richtung" laufen.

– Bei derart komplexen Entwicklungen ist das Testen allein mit Access oder Excel nicht möglich. Der hier zum Schluss gewählte Schritt wäre von vornherein der sinnvollste Ansatz gewesen.

– Zur Plausibilisierung der Testergebnisse ist eine enge Abstimmung mit der Fachseite sowie sinnvollerweise auch Zugriff auf die Frontoffice-Systeme notwendig, um Fehlinterpretationen der Testergebnisse zu vermeiden.

– Es ist besonders wichtig, die abgestimmte Testvorgehensweise und -struktur (inklusive Dokumentationsmethodik beziehungsweise Testprozess) während des gesamten Projektes beizubehalten, da Tests bei agilen Projekten mit weitgehender Flexibilisierung des fachlichen und technischen Vorgehens nicht selten eine einzige Basis für die notwendige Stabilität für Projektaktivitäten bilden. Dies bedeutet nicht, dass notwendige und wesentliche Anpassungen nicht zugelassen werden können – vor der Realisierung sollten sie aber kritisch auf mögliche Konsequenzen (zum Beispiel Auswirkung auf die Transparenz beziehungsweise Revisionsfähigkeit) überprüft werden. Gefährlich sind vor allem kleine, häufige Änderungen, die rein aus Effizienzgründen die Nachvollziehbarkeit und Klarheit der Testmethodik, der Prozesse sowie der Dokumentation gefährden können.

Potenzielle Problemfelder

Im Rahmen der Lessons Learned muss auf ein paar potenzielle Problemfelder hingewiesen werden. Dazu gehören:

– Das Tempo der Entwicklung und der Zwang der Einhaltung terminlicher Ziele machen eine vernünftige Releasepolitik häufig unmöglich. Änderungen in der Software/(-komponenten) werden oftmals nicht dokumentiert und können somit nicht nachvollzogen werden, was wiederum methodisches Testen nahezu unmöglich macht.

– Das zugrunde liegende Fachkonzept unterliegt häufig weiteren Änderungen. Es kommen neu Inhalte hinzu beziehungsweise alte Inhalte werden aufgrund neuer Erkenntnisse und Anforderungen angepasst. Das sich im Endeffekt ständig ändernde Fachkonzept ruft entsprechende Änderungen im Katalog der Testobjekte hervor. Die Einrichtung eines Prozesses für die Pflege des Katalogs ist keine leichte Aufgabe. Sie wird von Projekt zu Projekt anders aussehen und häufig einen informellen Charakter haben.

Generell wird festgestellt, dass der Testaufwand in agilen Projekten im Vergleich zur Wasserfallmethodik nahezu verdoppelt wird. Gleichzeitig werden die Gesamtziele des Implementierungsprojektes schneller erreicht, da viele Aufgaben in Teilprojekten durch agile Ansätze deutlich verkürzt werden konnten. Der schwer zu klassifizierende Vorteil des Projektes ist der deutliche Qualitätsgewinn, da die fachlichen Auftraggeber des Umsetzungsprojektes bestätigen, dass die neu zur Verfügung gestellte Funktionalität des Treasury-Systems den Bedürfnissen der Treasury-Verantwortlichen optimal angepasst ist.

Fußnoten

1) Das Wasserfallmodell ist linear und nicht iterativ, hat vordefinierte Start- und Endpunkte mit eindeutig definierten Ergebnissen und setzt sich aus folgenden Phasen zusammen: Anforderung, Entwurf, Implementierung, Tests, Wartung.

2) „Testen in agilen Projekten unterscheidet sich vom klassischen Testen in erster Linie dadurch, dass dieselben Tests viel häufiger ausgeführt werden müssen. Schließlich wird das System immer wieder geändert (Refactoring) und viel häufiger ausgeliefert. Daher lohnt sich die Automatisierung der Tests in agilen Projekten viel früher als in klassischen Projekten." Quelle: http://www.it-agile.de/schulung/agiles-testen-und-architektur

3) „Agile Projekte können auch massive Probleme im klassischen Testvorgehen verursachen: Detailspezifikationen sind erst (wenn überhaupt) kurz vor der Implementierung verfügbar und der Test soll gleichzeitig mit der Entwicklung am Ende jeder Iteration abgeschlossen werden. Bei Iterationslängen von wenigen Wochen verursacht das beträchtlichen Mehraufwand für den Test, der sich noch dazu am Ende der Iteration konzentriert, wodurch das Ziel eines voll getesteten Systems am Ende jeder Iteration oft nicht erreicht werden kann." Quelle: http://www.techtalk.ch/Agile-Business/Training-Agiles-Testen-und-Spezifikation-mit-Beisp.

4) „In agilen Projekten werden Tests als ausführbare Spezifikationen verstanden. Wie in klassischen Projekten auch unterscheidet man in agilen Projekten zwischen der technischen Spezifikation in Form von Unit-Tests und der Spezifikation der Anwendungsdomäne in Form von Akzeptanztests." Aus http://www.it-agile.de/schulung/agiles-testen-und-architektur.

5) Testobjekt: Eine Einheit (oder unit) bezeichnet (...) je nach zugrundeliegendem Programmierparadigma eine einzelne Prozedur, Funktion oder Methode, in: Norbert Oster, „Automatische Generierung optimaler struktureller Testdaten für objekt-orientierte Software mittels multi-objektiver Metaheuristiken", Dissertation an der Universität Erlangen-Nürnberg.