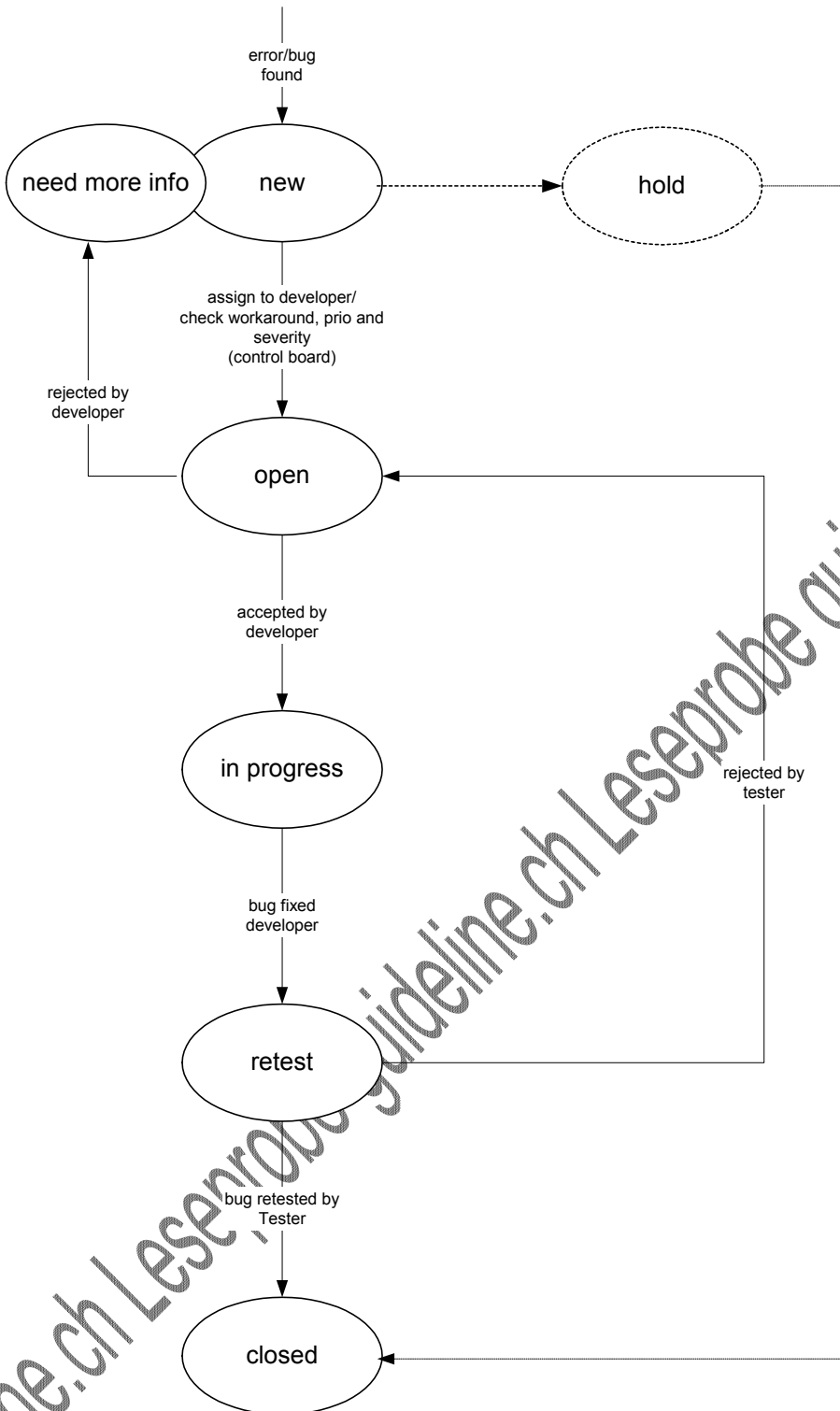


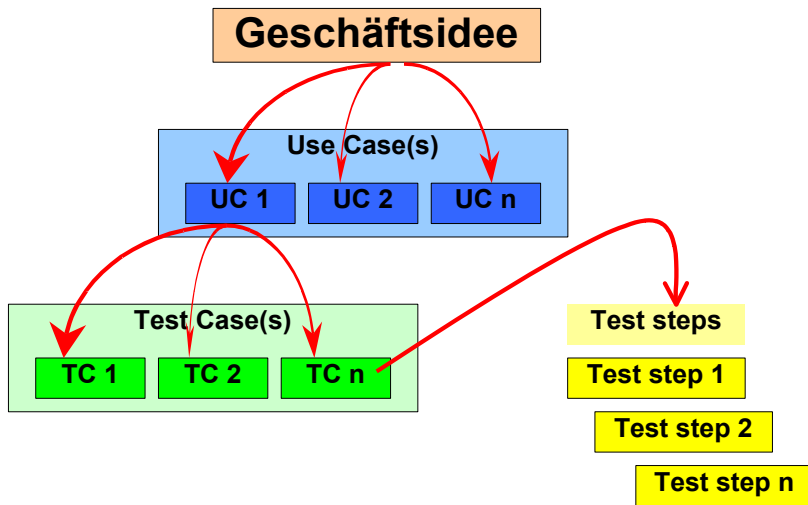
Tabelle 9: Defect Tracking Musterprozess



Nicht aufgeführt in dieser Darstellung ist zum Beispiel die Umwandlung eines gefundenen Fehlers in einen Change Request (CR).

Hier ein Beispiel eines Namenskonventionsmodell:

Tabelle 11: Namenskonventionsmodell



Die Umsetzung würde dann etwa so aussehen :

Tabelle 12: Namenskonventionsbeispiel

Was	Beschreibung
UC001_Kleinkreditrechner	Namensgebung für den Use Case/Applikation Kleinkreditrechner. Unter einem Use Case ist ein kompletter Anwendungsfall zu verstehen.
TC001_GrenzwertOben oder TC001_001_GrenzwertOben oder TC001_001_Grenzwert_Oben	Namensgebung für den Testcase „oberer Grenzwert“ des Kleinkreditrechner
TC001_GrenzwertOben_UT	Namensgebung für den Testcase „oberer Grenzwert“ des Kleinkreditrechner, zugeordnet zur Teststufe UNITTEST

Sie sehen, die Namensgebung kann beliebig fein sein.

2.5.1 Systemtest Prozessfluss

Ausgeführt werden diese Tests auf dem Testsystem 1 oder 2 oder auf dem Produktionsbackupsystem. Dies ist ein Musterprozess. Für die Implementation in der Praxis sind Anpassungen unumgänglich. Diese Teststufe wird oft in einem Testcenter durchgeführt, d.h. mit der aufgeführte Rolle des Testers ist nicht mehr der Projekttester, sondern der Tester des Testcenters gemeint, also eine „unabhängige“ Instanz.

Tabelle 23: Systemtest (Subprozess)

System Test

	Beschreibung	Rolle	Dokumente
<pre> graph TD Start([Start]) --> ECK[Eintrittskontrolle] ECK --> D1{o.k. / Nein} D1 -- Nein --> ECK D1 -- o.k. --> IDI[Installation/De-installation (Fallback)] IDI --> D2{o.k. / Nein} D2 -- Nein --> IDI D2 -- o.k. --> KT[Konfigurationstest] KT --> D3{o.k. / Nein} D3 -- Nein --> KT D3 -- o.k. --> RT[Regressionstest] RT --> D4{o.k. / Nein} D4 -- Nein --> RT D4 -- o.k. --> ST[Securitytest] ST --> D5{o.k. / Nein} D5 -- Nein --> ST D5 -- o.k. --> BRT[Backup & Recovery Test] BRT --> D6{o.k. / Nein} D6 -- Nein --> BRT D6 -- o.k. --> FT[Funktionale Tests] FT --> D7{o.k. / Nein} D7 -- Nein --> FT D7 -- o.k. --> Ende([Ende]) </pre>	<p>Prozess: Systemtest IN-Event: Unit Integrationtest abgeschlossen</p>		
	Formale Kontrolle der Anlieferung auf Struktur und Vollständigkeit. Dokumentenkontrolle.	Tester	< Release (Code & Dokumentation) < Test Case/Class < Dokumentation > Freigabe für Systemtest
	Installation gemäss Installationsanleitung. Fallback Kontrolle durch De-installation.	Tester Betreiber	< Release de-installiert > Release in- & de-installierbar
	Erneute Installation und Kontrolle der Konfiguration/Verträglichkeit.	Tester Betreiber	< Release installiert > Konfiguration kompatibel
	Kontrolle ob alte Fehler behoben sind und ob bereits getestete Funktionen noch lauffähig sind.	Tester Benutzervertreter	< Release > Release regression getestet
	Sicherheitsüberprüfung der Datenhaltung und der Datenzugriffe.	Tester Sicherheitsverantwortlicher	< Release < Sicherheitsanforderungen > Release sicherheitsüberprüft
	Sicherungs- und Fehlerverhalten muss wie spezifiziert funktionieren (Backup Test, Recovery Test, Fail Over Test).	Tester Betreiber	< Release > Backup getestet > Recoverytest > Fail Over Test
	Eingeschränktes Durchspielen von Testfällen anhand der Testfallbeschreibung. Fokus liegt auf UAT.	Tester Benutzervertreter	< Release < Spezifikation < Funkt. Testfälle > Release funktional getestet
	Prozess: Systemtest OUT-Event: Systemtest durchgeführt		

Tipps

- Versehen Sie diese Teststufe mit einer formalen Eingangskontrolle
- Testen Sie unter Einbezug des späteren Anwendungs- oder Systembetreibers
- Prüfen Sie, ob bekannte Fehler behoben wurden
- Prüfen Sie, ob sich keine neuen Fehler eingeschlichen haben (Regressionstest)
- Prüfen Sie, alle betrieblich relevanten Aspekte

-
- Prüfen Sie die Einführungsart auf einem Testsystem (Stichtag, Stufenweise, Parallelbetrieb)

2.6 Load- & Performancetest (Subprozess)

Definition : Teststufe. Tests, die den Nachweis zum Ziel haben, dass das System (Hard- & Software) die vom Auftraggeber geforderte Leistung, sowie eine angenommene Spitzenbelastung zur Zufriedenheit bewältigt. Getestet wird durch Veränderung der Last bei gleichbleibender Konfiguration und zusätzlich bei gleichbleibender Last durch Veränderung der Konfiguration. Als Nebenprodukt entsteht eine Basislinie für das Monitoren im späteren Betrieb.

Kernaussage 1 : Für eine Web Applikation ist diese Teststufe unverzichtbar !

Kernaussage 2 : Da Sie für diese Teststufe sehr viel Geld aufwenden, ist eine minutiöse Planung unerlässlich !

Kernaussage 3 : Ziehen Sie erfahrene Leute zu Rate.

An dieser Stelle entstehen oft Diskussionen mit Fachleuten, die sagen: „Aber Performancetest ist doch keine Teststufe, sondern eine Testart, die innerhalb des Systemtests durchgeführt wird !“ Prinzipiell ist diese Aussage völlig richtig, aber mit dem Wandel in der IT hin zu Netzanwendungen wird dieser Test immer aufwendiger und immer wichtiger. Ausserdem ist das benötigte Know-how für die Vorbereitung und Konzeptionierung, sowie für die Durchführung mit Testwerkzeugen nur sehr rar vorhanden. Fast alle anderen Testaktivitäten können von gelernten Berufsleuten (Entwickler/Architekten/System-Administratoren etc.) durchgeführt werden, bei Performancetest hört das „Latenspiel“ aber auf. Hier brauchen Sie auf diesen Testbereich spezialisierte Profis. Ein Performancetest ist eine eigenständige Teststufe im Testprozess. Diese Teststufe wird vor dem Benutzerakzeptanztest durchgeführt, weil die Abnahme sinnlos ist, wenn das System nicht dem geforderten Mengengerüst standhält.

Wenn Sie sich fragen, ob diese Testart nötig ist, dann erinnern Sie sich daran, dass speziell bei e-commerce Anwendungen die Zahl der Benutzer explodieren kann. Das Web ist potentiell Ihre grösste Filiale und ein Ausfall betrifft sofort eine grosse Anzahl Ihrer Kunden. Betreiben Sie also den Aufwand entsprechend.

Laut dem Rational Unified Process (RUP) Framework zerfällt der Performancetest in :

- **Benchmark Testing**; Vergleicht die Performance eines neuen Tests (noch nie durchgeführten) gegen ein bestehendes System(-teil).
- **Contention Test**; Verifiziert, dass eine Systemressource eine adäquate Anzahl von Requests abarbeiten kann. z.B. 30 DB Zugriffe in 10 Sekunden.
- **Performance Profiling**; Bei gleichbleibender Belastung wird die Konfiguration des Systems verändert, um zu sehen, ob Verbesserungen eintreten.
- **Load Testing**; Bei gleichbleibender Konfiguration des Systems wird das Belastungsprofil (Anzahl Benutzer, Anzahl Transaktionen) verändert, um zu sehen, ob Veränderungen eintreten.

- **Stress Testing;** Bei gleichbleibender Konfiguration des System wird das Belastungsprofil (Anzahl Benutzer, Anzahl Transaktionen) anormal angehoben, um zu sehen, wann das System zusammenbricht.

Für die Umsetzung erstellen Sie zuerst ein Konzept welches folgende Punkte klärt:

- Welche Tests werden durchgeführt (Metriken, Anforderungen)
- Auf welchem System werden die Tests durchgeführt
- Unter welchen Bedingungen werden die Tests durchgeführt
- Wer führt die Tests durch
- Wer ist mit einzubeziehen
- Welche Werkzeuge werden eingesetzt
- Testdatenbereitstellung
- Testdokumentation

„A problem is not always located where it is found“

“The solution to a problem is not always located where the problem is found”

2.6.1 Werkzeuge

Obwohl bis vor kurzem noch Lasttests in einer mitteleuropäischen Versicherung mittels Megaphon Zuruf durch mehrere hundert Studenten gemacht wurde, stehe ich zu der Aussage „Ohne Werkzeuge geht es nicht mehr“.

2.6.2 Skripts

Code der entsteht, wenn Sie mit einem Werkzeug einen definierten Use Case (oder Business Case) aufzeichnen. Werden von einem virtuellen User mit Hilfe des Werkzeuges abgespielt. Die meisten Werkzeuge verwenden eine übliche Programmiersprache wie C, C++ und können somit durch Entwickler modifiziert werden. Zum Beispiel kann ein Eingabewert durch eine Variable ersetzt werden, die aus einer Tabelle oder Datenbank gelesen wird, um so das Caching Problem zu umgehen.

2.9 Pilot Test (Subprozess)

Definition : Teststufe. deu. Feldtest. Produktive Ausbreitung der Lösung auf einen beschränkten Bereich der Unternehmung. Teststufe ist nicht in jedem Fall sinnvoll.

Die Pilot Teststufe ist eine „Sicherheitsbarriere“ vor der Produktion. Die Durchführung dieser Stufe ist nicht in jedem Fall angebracht und zweckmässig. Am besten eignen sich grosse, unternehmensweite applikatorische Projekte, bei denen die Möglichkeit einer initialen Ausbreitung in eine Filiale oder eine Abteilung besteht. Auch bei Hardware-Beschaffungen in grosser Stückzahl ist die Pilot-Bestückung eines oder mehrerer Arbeitsplätze sinnvoll. Fällt eine neue applikatorische Ausbreitung mit einer neuen Hardwareausbreitung zusammen, so ist ein Pilot sehr empfehlenswert.

Die Pilotphase soll die Alltagstauglichkeit einer Applikation oder Hardware beweisen. Es ist wichtig, dass dies geplant geschieht, um sicher zu stellen, dass die ganze Funktionalität abgedeckt wird.

Testart	Kurzbeschreibung
Tagesgeschäft (Funktionale Tests)	Tagesgeschäft in begrenztem System mit eingeschränktem Benutzerkreis Basis: Testdrehbücher oder Use Case-Beschreibungen
Business Continuity	Die im Business Continuity Plan aufgeführten Anforderungen können erfüllt werden. Eventuell wird eine realistische Übung durchgeführt.
Mögliche weitere Tests, wobei es sich hierbei auch um Wiederholungen handeln kann:	
Securitytest	Zugriffskontrollen, Übertragungsvorgänge etc. durch Spezialisten.
Dokumentation	Kontrolle der mitgelieferten Dokumente auf Vollständigkeit, Fehler.
Testart	Kurzbeschreibung
Backuptest	Ist ein Wartungsfenster definiert, kann dieses mit den geforderten Volumen eingehalten werden.
Recoverytest	Ist das Verhalten nach einer Überlast wie gewünscht und beschrieben.
Fail Over Test	Übernimmt das zweite System oder die zweite HD, oder das RAID Set bei einem Ausfall.
End of Period Test	Tages-, Wochen-, Monats-, Quartals- und Jahresendverarbeitung mit entsprechenden Testdaten.
Linktest	Auf Inter- oder Intranetportalen werden alle aufgeführten Links getestet. Automatisierungspotential.
Loadtesting	Belastung des Systems mit einer progressiven Anzahl Zugriffe, die vorher vom Auftraggeber oder dessen Stellvertreter festgelegt wurden. Inkl. Contention Tests.
Stresstest	Belastung des Systems mit einer anormalen Anzahl Zugriffe, die vorher vom Auftraggeber bestimmt wurde. Meistens 10 x die höchst erwartete Anzahl Zugriffe.
Volumentesting	Meist als Unterteil des Loadtesting. Die Datenbanken werden mit der erwarteten Anzahl Datensätze befüllt. Sind

	die Werte des Loadtesting auch noch mit befüllten Tabellen im erwarteten Bereich ?
Breaktest	Belastung des Systems mit einer anormalen Anzahl Zugriffe bis zum System-Crash.
Performance Profiling	Veränderung der Systemkonfiguration bei gleichbleibender meist durchschnittlicher Belastung.

2.9.1 Pilot Test Prozessfluss

Ausgeführt werden diese Tests auf dem Produktionsbackupsystem oder auf dem Produktionssystem. Dies ist ein Musterprozess. Für die Implementation in der Praxis sind Anpassungen unumgänglich. Je nach dem, wie viel vom Alltagsgeschäft Sie austesten wollen, erhöhen sich die Testarten und eventuell auch die Teststufen.

Tabelle 29: Pilot Test (Subprozess)

Pilot Test

	Beschreibung	Rolle	Dokumente
<pre> graph TD Start([Start]) --> Tagesgeschäft[Tagesgeschäft] Tagesgeschäft --> D1{o.k.} D1 -- Nein --> Tagesgeschäft D1 -- o.k. --> BC[Business Continuity] BC --> D2{o.k.} D2 -- Nein --> BC D2 -- o.k. --> Z4[Zusatztest 4] Z4 --> D3{o.k.} D3 -- Nein --> Z4 D3 -- o.k. --> Z2[Zusatztest 2] Z2 --> D4{o.k.} D4 -- Nein --> Z2 D4 -- o.k. --> Ende([Ende]) </pre>	<p>Prozess: Pilot Test IN-Event: User Acceptance Test durchgeführt</p> <p>Tagesgeschäft in begrenztem System mit eingeschränktem Benutzerkreis Basis: Testdrehbücher oder Use Case Beschreibungen</p> <p>Wiederaufnahme der Geschäftstätigkeit nach einem Katastrophenfall gemäss Business Continuity Plan</p> <p>Wiederholung von vorherigen Test, ev. mit anderen Testdaten und durch andere Benutzer.</p> <p>Wiederholung von vorherigen Test, ev. mit anderen Testdaten und durch andere Benutzer.</p> <p>Prozess: Pilot Test OUT-Event: Pilot Test durchgeführt</p>	<p>Entwickler (Support durch Tester)</p> <p>Betreiber Benutzervertreter</p>	<p>< Requirments < Code < Programmier-richtlinien > Tested Code > Test Case/Class > Dokumentation</p> <p>< Business Continuity Plan ungeprüft < Business Continuity Plan geprüft</p>

❗ Tipps

- Planen Sie die Dauer der Phase
- Schulen Sie die Pilot Benutzer
- Legen Sie die Prüfkriterien der Pilotphase fest
- Prüfen Sie die Dokumentation
- Testen Sie unter Einbezug des Betreibers
- Testen Sie unter Einbezug der späteren Supportorganisation
- Prüfen Sie die Supportorganisation wie Helpdesk, Onsite Support etc.
- Organisieren Sie eine Abschluss-/Freigabesitzung
- Erstellen Sie einen Schlussbericht

2.10 Minimalvariante des Testprozesses

Definition : Prozess. Festlegung der Tätigkeiten zur Qualitätssicherung, die als absolutes Minimum für eine „akzeptable“ Erreichung der Qualität gelten.

Während der Erstellung dieses Buches bin ich mehrmals von Korrekturlesern gebeten worden, doch einmal aufzuzeigen, wie die Minimalvariante eines Testprozesses aussehen könnte.

Dies hier ist nun der Versuch, diesem (Kunden-)Anliegen nachzukommen. Aufgrund der Vielzahl von Projekttypen und Arten komme ich diesem Wunsch mit einem unguuten Gefühl nach.

Kernaussage 1: Minimierung der Testarten und Testfälle erhöht Risiken.

Kernaussage 2: Setzen Sie Kosten nicht verfremdet ein. Das heisst, veranschlagte Testkosten werden auch für das Testen verwendet und nicht für die Aufstockung von schlecht kalkulierten Budgetposten.

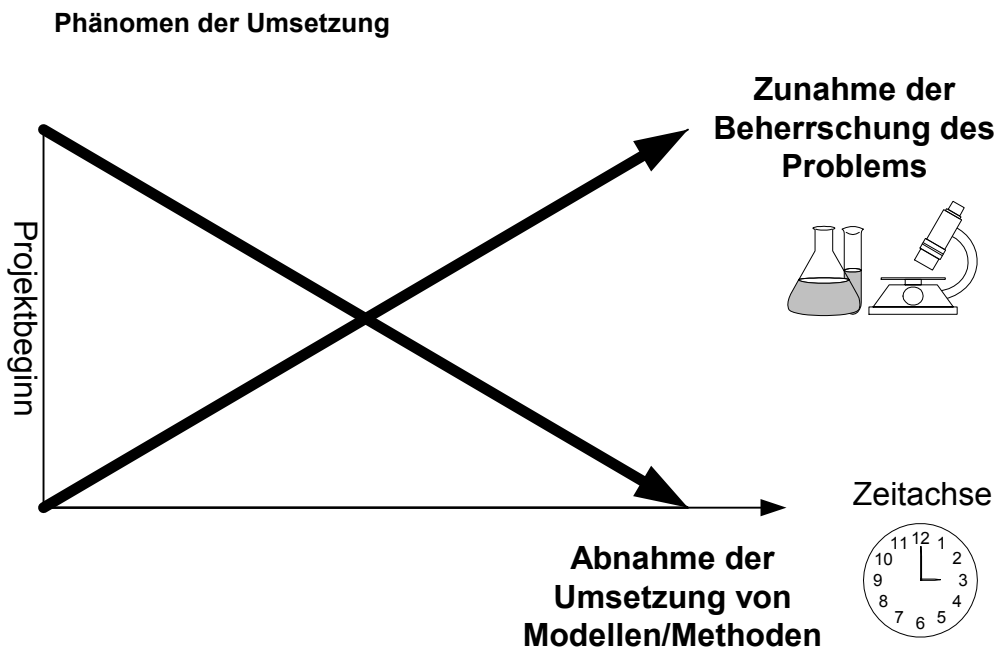
Kernaussage 3: Wurden die Kosten falsch veranschlagt, so analysieren Sie warum und speisen Sie Ihre Knowledge Datenbank (KPI's). Beschäftigen Sie sich mit den Schätzmethode und spezialisieren Sie sich auf eine. Ziehen Sie erfahrene Leute zu Rate.

Kernaussage 4: Schreiben Ihr Projektmanagementprozess kein Controlling vor, so initialisieren Sie es selber.

Unterhält man sich mit IT-Projektleitern aus allen Bereichen und fragt nach dem Budgetanteil für das Testen und die Qualitätssicherung, so erhält man Antworten die sich im Bereich von 30-60 % des Gesamtbudgets bewegen. Bei einem Mittelwert von 45 % und einem Gesamtbudget von € 1'000'000 sind das stolze € 450'000. Richtig eingesetzt, müsste kein Projektleiter bei diesem Betrag nach einem „minimalen Testprozess“ fragen.

Die Kernaussage 2 würde grafisch dargestellt etwa so aussehen:

Tabelle 30: Phänomen der Umsetzung



Dieses Verhalten können Sie in annähernd 100 % aller IT-Projekte beobachten. Warum ist das so ? Es sind wahrscheinlich Faktoren wie, nicht Beherrschen des Sachgebietes, fehlende oder nicht beherrschte Methoden, falsche Einschätzungen, Unvorhergesehenes, Torschlusspanik, falsches Kundenorientierung usw. Schuld daran. Dieser Umstand ist für viele selbstverständlich. Auf den Flugbetrieb umgelegt, wäre das in etwa so, als ob der Pilot vor dem Start eine paar Checks nicht ausführt, weil die Passagiere (Kunden) sich schon über die fünfzehn Minuten Verspätung aufregen.

Was lässt sich dagegen machen ? Die Antwort könnte heißen „Rückgrat“ ! Das heisst, wenn die Methode bekannt ist, dann sollten Sie sie anwenden, und sich nicht durch „falsche Hektik“ ablenken lassen. Wenn die Methode nicht bekannt ist, dann legen Sie sich eine zu. Den ersten Schritt haben Sie ja bereits mit dem Lesen dieses Buches getan. Wenn Sie es aber lesen und nicht alles oder gebrauchte Teile davon konsequent umsetzen, dann sind Sie gleich weit wie vorher. Genau genommen ist es sogar ein Rückschritt, denn Sie haben ja für Nichts Zeit und Geld investiert.

Was sind die Tests (Testarten und Teststufen), die ich mindestens tun muss ? Wenn Sie dieses Buch bis hierher gelesen haben, so prüfen Sie einfach jede Testart mit der Kontrollfrage: „Kann ich es mir leisten, diesen Test nicht zu machen?“ Was danach übrig bleibt, ist.... „IHR minimale Testprozess“.

2.10.1 Minimalvariante Prozessfluss

Ausgeführt werden diese Tests auf dem Tests-, Produktionsbackup- oder Produktionssystem. Dies ist ein Musterprozess. Für die Implementation in der Praxis sind Anpassungen unumgänglich.

Tabelle 33: Verantwortlichkeiten der Rollen (Beispiel)

<div style="display: flex; justify-content: space-between;"> \ Rolle </div>	Entwickler	Chefentwickler	Tester	Test Tool Engineer	Testmanager	Benutzervertreter ①	Betreiber ①	Release Manager ①	Konfigurationsman. ①	Qualitätsmanager ①	Auftraggeber ①	andere
Lieferobjekt												
Testvorbereitung												
Prozessimplementation					●					●		
Testplanung			●		●			●				
Releaseplanung								●				
Testdaten			●		●						●	
Tool-Installation				●								Tool-Consultant
Tool-Schulung				●								
Testvorbereitung			●		●							Systemadministrator
Testdurchführung												
Unit Test	●	●										
Unit Integrationstest	●	●										
Systemtest					●		●					
Performancetest			●		●							
User Acceptance Test					●	●					●	
Pilot					●	●	●	●	●			
Einführung						●	●	●	●		●	
Testauswertung												
Testauswertung		●	●		●					●		
Change-Kategorisierung		●			●	●	●			●	●	durch Gremium
Defect-Kategorisierung		●			●	●	●			●	●	durch Gremium
Defect Tracking		●			●							
Release-Freigabe					●			●				
Statische Tests					●					●		

● = Gesamtverantwortung, ● = Mitarbeit / Input, ① = Rolle wird vom Kunden wahrgenommen bei Auftragsprojekten an Drittfirmen

3.2 Schulung im Testteam

Die Einarbeitungsphase von neuen Testteammittgliedern ist sehr wichtig. Erhalten Sie Ihre Testteammittglieder nicht von einem Test Competence Center, so müssen Sie auch einen gewissen Aufwand für die Schulung & Weiterbildung betreiben. Zu schulen sind aufgesetzte Prozesse und Werkzeuge.

Stellen Sie einem neuen Teammitglied einen Mentor für die Anfangsphase zur Verfügung. Bei grossen Projekten oder als Teil des Projektmanagement-Framework kann auch ein sogenannter „Projekt Knigge“ erarbeitet werden, der die wesentlichsten Spielregeln und die Interaktion im Team und im Umfeld des Projekts regelt.

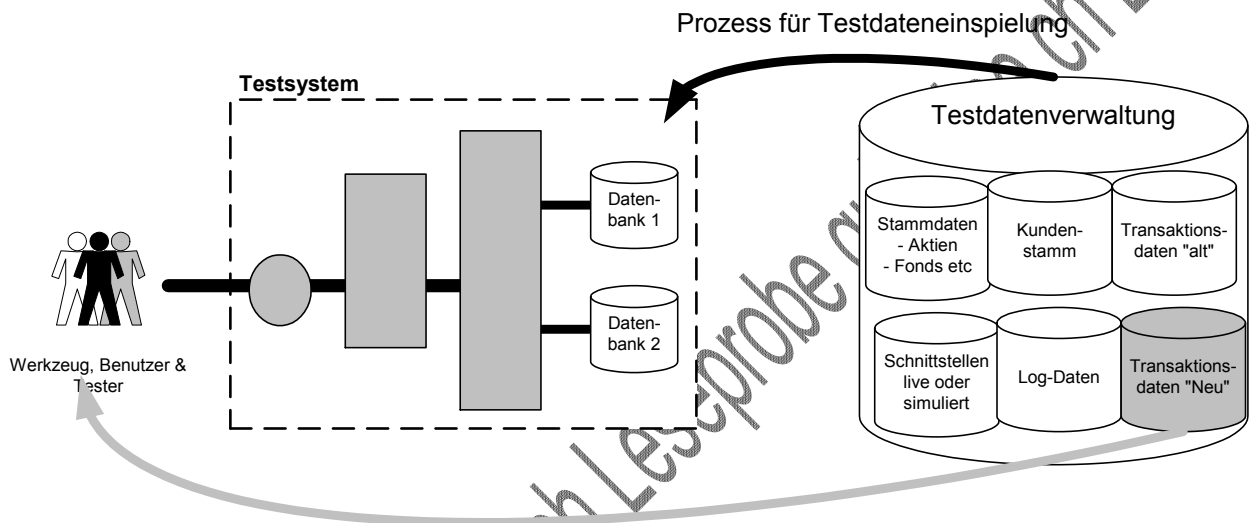
Führen Sie wenn immer möglich „on the job“ Schulungen durch. Externe Schulung sind oft zu abstrakt und das Erlernete muss noch auf das konkrete Problem projiziert werden.

5. Testdaten

Kernaussage : Wo Dateneingabe nötig ist, werden Testdaten gebraucht. Input in Form von Navigationsanweisungen werden in den Testfällen beschrieben und sind keine Testdaten.

Wie vielschichtig das Thema Testdaten ist, zeigt Ihnen diese Grafik. Die Grafik zeigt auch, dass Testdatenpflege und Bereitstellung Abhängigkeiten von und zu Konfiguration- und Release Management hat.

Tabelle 39: Testdaten



Testdatenmanagement ist wohl die meist unterschätzte Disziplin beim Testen von Software. Oft wird der Aufwand für das Erstellen und Pflegen von Testdaten völlig unterschätzt. In einem umfangreichen e-commerce-Projekt habe ich erlebt, dass mehr als zehn Leute für das Datendesign zuständig waren, aber gerade nur eine Person für die Testdaten. Das heisst, eine Person versucht die gleiche Arbeit zu tun wie es die anderen zehn Entwickler tun.

Will man ein komplettes Banking-System testen, so braucht man Kunden, diese Kunden brauchen Konten (in diverse Währungen) und Portfolios (in diverse Währungen), diese Kunden brauchen eine Kreditlimite, ein Rating (Kundenbewertung), einen Kundensachbearbeiter, eine Filiale, ein Verhaltensprofil.

Tätigt dieser Kunde nun einen Aktienkauf, so muss zuerst das Konto mit Guthaben vorhanden sein, dann muss zumindest ein leeres Portfolio vorhanden sein. Will man eine Pan-Europäische Bank mit 500'000 Kunden testen, dann stehen Sie vor einer Herausforderung.

Aufgrund dieser Transaktionsdaten möchten Sie dann alle „End of“-Jobs durchspielen und die Daten verifizieren.

Zugleich meldet das MIS (Management Information System) einen Bedarf zum Testen der spezifizierten Reports an. Diese Tests, das heisst die Datenzustände zu gewissen

Zeitpunkten müssen Sie in Ihrer Planung berücksichtigen, da meist weder genügend Zeit noch Arbeitskraft vorhanden ist, damit Sie diese Tests separat durchführen können.

Dieses Beispiel, das zugegebenermassen nicht ganz alltäglich ist, zeigt die Dimensionen in denen Sie sich als Testdatenverwalter bewegen. Und wir haben noch nicht von Fall-Back-Szenarien, Hot Swapable Systems und Load Balancing geredet !

Grenzwerttests für sich alleine genommen sind eigentlich nicht sehr anspruchsvoll. Wenn alles grösser und gleich hundert gültig ist es relativ banal. In einer Verkettung von Abhängigkeiten für eine Berechnung steigt die Komplexität aber sprunghaft an.

Ein Beispiel:

1. Grenzwertberechnung; Leasingobjekt => € 100'000
aber
2. Grenzwertberechnung; Verschuldungsgrad des Kunden =< 10 % des brutto
Monatseinkommen
3. Grenzwertberechnung; Anzahl bestehender Leasingverträge =< 3

dies ergibt folgende minimalen Testdatensätze:

positives Resultat des Durchlaufs:

- € 100'000, 10 %, 3
- € 101'000, 9 %, 2

negatives Resultat des Durchlaufs:

- € 99'000, 10 %, 3
- € 100'000, 11 %, 3
- € 100'000, 10 %, 4

Für diesen Test werden üblicherweise zwei Testfälle erstellt:

Leasing-Offerte erfolgreich erstellt

und

Leasing-Offerte abgelehnt

Die Testdaten wie sie oben aufgeführt sind, werden für beide Testfälle separat gehalten, zum Beispiel in je einer MS Excel ® Datei. Dies hat den Vorteil, dass die Varianten für die Durchführung bei einer späteren Automatisierung jederzeit erweitert werden können. Warum werden diese Testdaten nicht gemeinsam gehalten ? Dies hat zwei Gründe, erstens kann der Programmablauf nach einer Fehler- oder Warnmeldung erheblich von einem Normalablauf abweichen. Zweitens würden Ihre Testanalyse und Ihre Kennzahlen verfälscht, denn ein Testfall, von dem erwartet wird, dass er die Offerte ablehnt und dies auch tut, ist positiv. Ein Testwerkzeug zählt diesen aber, wenn er mit dem normalen Testfall generiert wurde, als Fehler bzw. negativ.

Wenn Sie sich also klar sind, wie Sie die Testfälle aufteilen, dann geht es an die Testdatenerstellung oder die Testdatenbeschaffung. Ein möglicher Weg ist die manuelle Erstellung der Testdaten. Wenn dies nicht möglich ist, dann bleibt Ihnen noch die automatische Testdatenerstellung.

Es gibt eine neue Generation von Werkzeugen, die bei der automatischen Erstellung hilft. Hier eine nicht abschliessende Liste:

Hersteller	Quelle	Werkzeug
Data Tect	http://www.datatect.com/	Data Tect™
Computer Associates	http://www.cai.com/products/datamacs_ii.htm	CA-Datamacs/II
SKILL	http://www.skill.it/Eng/Software/System/Products/QTEST.html	QTEST
Quest Software	http://www.quest.com/datafactory/	Data Factory™
COMPUWARE	http://www.compuware.com/producte	File-Ald

① Tipps

- Der Datenkatalog (Datenmodell) muss die Basis für die Testdatenerstellung bilden !
- Verwenden Sie Werkzeuge
- Vermeiden Sie manuelle Datenerstellung !
- Gefahr der Fokussierung auf einzelner Use Cases besteht !
- Involvieren Sie verschiedene Rollen in die Testdatengenerierung
- Verwenden Sie Extrakte aus bestehenden Daten (verfremdete)
- Messen Sie die Testabdeckung des Datenmodells (Soll 100 %)
- Passen Sie die Grösse der Testdatensätze der Applikation an (mehr ist besser) !
- Trennen Sie positive und negative Testdaten
- Führen Sie Grenzwerttestdaten separat

5.1 Testdaten aus produktiven Daten

Dies ist bei weitem nicht so simpel, wie es der Titel vermuten lässt. In den wenigsten Fällen, können Sie einfach eine Kopie von einem produktiven Datenbestand ziehen und auf Ihrem Testsystem einspielen. In vielen Fällen gibt es neben organisatorischen, firmenpolitischen auch noch rechtliche Aspekte zu berücksichtigen.

Es gilt, Testdaten aus produktiven Datensätzen dürfen keine reale Zuordnung von Geschäftsvorfällen zu real existierenden Kunden mehr zulassen. Deshalb müssen die Daten so verfremdet werden, das kein „Stein mehr auf dem anderen“ bleibt.

Kunden #	Vorname	Nachname	Strasse	Ort	
1	Hans 1	MXuster 1	Strasse No.1	Ort 1	
2	Neue Nummern	Hans 2	MXuster 2	Strasse No.2	Ort 2
3		Hans 3	MXuster 3	Strasse No.3	Ort 3
4	Hans 4	MXuster 4	Strasse No.4	Ort 4	
5	Hans 5	MXuster 5	Strasse No.5	Ort 5	

Danach können noch einzelne Buchstaben ersetzt werden, zum Beispiel jeder zweiter Buchstabe im Feld Nachname durch ein X usw.

5.2 Testdaten für Load- & Performancetests

Die Bereitstellung der Load- & Performancetestdaten stellt wieder etwas andere Anforderungen als die normalen Testdaten. Die Daten müssen nur noch bedingt „logisch“ sein, dafür steigt die Menge der Daten an. Bedingt logisch heisst, das der Ablauf nicht durch unlogische Daten gestoppt werden darf. Zum Beispiel, wenn ich aus

nicht funktionale Punkte. Für Projekte ist die Qualität der Anforderungsspezifikation ein Schlüsselerfolgswert.

APM

→ allgemein
 eng. application performance management
 Überwachen der Verfügbarkeit von geschäftskritischen Anwendungen in einem Unternehmen.

Application Infrastructure Providing, siehe **AIP**

Application Performance Management, siehe **APM**

Application Service Provider, siehe **ASP**

Application Service Providing, siehe **ASP**

Approval

→ allgemein
 deu. Billigung
 Schriftliche Bestätigung einer autorisierten Person, dass Code, Dokumentation, Pläne oder Ähnliches den Vorgaben und Erwartungen entspricht. Nach dem Approval können die geplanten, weiteren Schritte eingeleitet werden.

Äquivalenzklasse

→ allgemein
 eng. -
 Eingabewertesammlung, die zu einer gleichen (identischen) Verarbeitung führen.

Architekt

→ Rollen im Testprozess
 eng. architect, designer
 Definition der Systemarchitektur (logische und physische). Der Architekt legt die verwendeten Technologien und deren Zusammensetzung fest, die zur Erreichung der Auftragsziele notwendig sind.

Architektur

→ allgemein
 eng. architecture, system layout
 Meint den Aufbau, die Zusammensetzung des Systems bzw. auch dessen graphische Darstellung. Dient für die Übersicht und die Erkennung der Gesamtzusammenhänge. Hilfreich für die Abgrenzung und Aufteilung der Testaktivitäten.

Artefakt

→ allgemein
 eng. artifact
 Lieferobjekte (Lieferergebnis) im Entwicklungsprozess, z.B. ein Testkonzept,

Testplan, Systemarchitektur etc. Ausdruck wird auch im RUP verwendet.

ASCII

→ allgemein
 eng. american standard code for information interchange
 Standard Übertragungscode um Kompatibilität zwischen Datenquelle und Datenempfänger sicher zu stellen.

ASP

→ allgemein
 eng. application service provider
 eng. application service providing
 Ein Dienstleister, der Anwendungssoftware vermietet, welche auf Servern in einem Rechenzentrum gehostet wird und auf die online, unabhängig vom Betriebssystem, zugegriffen werden kann. Der Aufwand für Installation, Update, Hardware Pflege etc. wird für den ASP Kunden reduziert. Die Technik „Application Service Providing“ wird ebenfalls mit der Abkürzung ASP bezeichnet, was zu Verwirrungen führt.

Assessment criteria, siehe **Prüfschärfe**

ATLM, siehe **Automated Test Life-Cycle Methodology**

Audit

→ IEEE 610.12 & IEEE 1028
 eng./deu.
 Unabhängiges Begutachten eines Arbeitsprodukts oder eines Sets von Arbeitsprodukten um die Einhaltung von Plänen, Spezifikation, Standards, Arbeitsanweisungen und Verträgen zu kontrollieren. „Beantwortet die Frage, „Tun wir Es richtig?“, also Fokus auf den Entstehungsprozess und nicht auf das Produkt.

Audit Trail, siehe **Time Stamp**

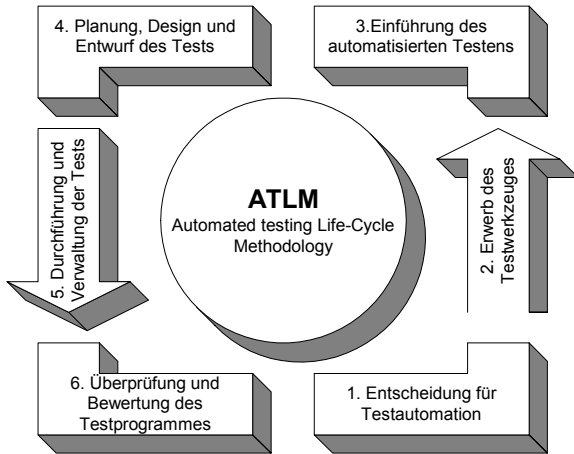
Auftraggeber

→ Rollen im Testprozess
 eng. client
 Erteilung des Auftrages und Bereitstellung der benötigten Ressourcen. Abnahme von Teil- und Endprodukten. Besetzung des Steuerungsausschusses.

Automated Test Life-Cycle Methodology (ATLM)

→ allgemein
 eng./deu.
 Strukturierte Methode, die auf eine frühzeitige Planung der Testaktivitäten im Projekt angewiesen ist und auf automatisierte Testwerkzeuge zur Unterstützung des Testprozesses setzt. ATLM lässt sich unterteilen

in 1. Entscheidung zur Automatisierung, 2. Erwerb des Testwerkzeuges, 3. Einführung des automatisierten Testens und 4. Planung, Design und Entwicklung der Tests. Siehe auch www.autotestco.com.



Availability

→ Testprozess

deu. Verfügbarkeit

Überbegriff für die Qualitätsmerkmale Zuverlässigkeit, Wartbarkeit und Servicegrad.

Availability Management

→ ITIL

deu. Verfügbarkeitsmanagement

Availability Management stellt sicher, dass aus der vorhandenen IT-Infrastruktur und den angebotenen Dienstleistungen ein maximaler Nutzen erzielt werden kann. Availability Management stellt die im SLA vereinbarten Leistungen sicher, bzw. überprüft diese.

B2B

→ allgemein

eng. business to business

Elektronischer Internethandel von Waren und Dienstleistungen zwischen Unternehmen, oft über B2B-Marktplätze.

B2C

→ allgemein

eng. business to consumer

Elektronischer Internethandel zwischen Unternehmen und Endverbraucher.

Baseline

→ IEEE610.12

deu. -

Eine Spezifikation oder Arbeitsprodukt, das abgenommen wurde bildet die „Messlatte“ für folgende Messungen. Eine Baseline und akzeptierte Änderungsanträge bilden die aktuelle Konfiguration. Im Zusammenhang mit Monitoring stellt die Baseline eine Messung im

Normalbetrieb da, anhand derer Anormalitäten beobachtet werden.

BCP, siehe **Business Continuity Planning**

Benchmarking

→ allgemein

eng./deu.

Prozess des Vergleichens und Messens von Produkten und Dienstleistungen. Es kann z.B. ein Produkt gegen ein anderes, eigenes oder ein Konkurrenzprodukt verglichen werden um so aus diesem Vergleich (mit Besserem) zu lernen. Unterschieden wird internes, wettbewerbsorientiertes und funktionales Benchmarking.

Benchmark Testing

→ RUP – Performance Testing

deu. -

Vergleicht die Performance eines neuen Tests (noch nie durchgeführten) gegen ein bestehendes System(-teil).

Benutzer

→ IEEE8402

eng. user

Person oder Personengruppe die direkt mit dem System im täglichen Betrieb interagiert. Benutzer und Auftraggeber sind meist nicht die gleiche Person.

Benutzerakzeptanztest

eng. user acceptance test (UAT)

Teststufe. High Level Tests der spätere Benutzer, bzw. delegierte Vertreter der Endbenutzer prüfen das Lieferobjekt gegen die Spezifikation. Hierfür werden Testfälle (mittels Test- oder Abnahmedrehbüchern) erarbeitet und durchgeführt. Fehler werden aufgenommen, priorisiert und dem Change-Management-Prozess zugeführt. Die Basis für die Testdrehbücher sind Use Cases bzw. Geschäftsfälle, wie sie in der Realität vorkommen. Das hierfür benutzte Testsystem muss dem produktiven Zielsystem so ähnlich wie möglich sein. Das Testteam nimmt eine begleitende, unterstützende Rolle ein.

Benutzerfreundlichkeit

→ V-Modell

eng. usability

Eigenschaft einer Software, die es erlaubt, die Aufwände für Einarbeitung oder Benutzung so gering wie möglich zu halten. Nach [ISO 9126] ist Benutzerfreundlichkeit ein Qualitätsmerkmal.

Es umfasst die Untermerkmale

- Verstehbarkeit (Understandability)
- Erlernbarkeit (Learnability)
- Handhabbarkeit (Operability)

Benutzervertreter, siehe **Power User**