Reihe: Telekommunikation @ Mediendienste · Band 6
Herausgegeben von Norbert Szyperski, Udo Winand, Dietrich Seibt, Rainer Kuhlen und Rudolf Pospischil

Martin Engelien/Jens Homann (Hrsg.)

Virtuelle Organisation und Neue Medien

Workshop GeNeMe99
Gemeinschaften in Neuen Medien

TU Dresden, 28./29.10.1999

JOSEF EUL VERLAG
Lohmar · Köln

Reihe: Telekommunikation @ Mediendienste · Band 6

Herausgegeben von Prof. Dr. Dr. h. c. Norbert Szyperski, Köln, Prof. Dr. Udo Winand, Kassel, Prof. Dr. Dietrich Seibt, Köln, Prof. Dr. Rainer Kuhlen, Konstanz, und Dr. Rudolf Pospischil, Brüssel

PD Dr.-Ing. habil. Martin Engelien
Dipl.-Inform. (FH) Jens Homann (Hrsg.)

Virtuelle Organisation und Neue Medien

Workshop GeNeMe99
Gemeinschaften in Neuen Medien

TU Dresden, 28./29.10.1999

JOSEF EUL VERLAG
Lohmar · Köln

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

GeNeMe <1999 Dresden> :
GeNeMe 99 : Gemeinschaften in neuen Medien ; Dresden, 28./29.10.1999, an der Fakultät Informatik der Technischen Universität Dresden / Technische Universität Dresden, Fakultät Informatik, Institut für Informationssysteme, Forschungsgruppe "Entwurfsmethoden und Werkzeuge für Anwendungssysteme". Martin Engelien ; Jens Homann (Hrsg.). – Lohmar ; Köln : Eul, 1999
(Reihe: Telekommunikation @ Mediendienste ; Bd. 6)
ISBN 3-89012-710-X

Josef Eul Verlag GmbH
Brandsberg 6
53797 Lohmar
Tel.: 0 22 05 / 91 08 91
Fax: 0 22 05 / 91 08 92
http://www.eul-verlag.de
eul.verlag.gmbh@t-online.de

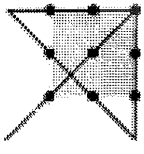
Printed in Germany
Druck: Rosch-Buch, Scheßlitz

Gedruckt auf säurefreiem, 100% chlorfrei gebleichtem, alterungsbeständigem Papier nach DIN 6738

Technische Universität Dresden
Fakultät Informatik • Institut für Informationssysteme
Forschungsgruppe „Entwurfsmethoden und Werkzeuge für Anwendungssysteme"

PD Dr.-Ing. habil. Martin Engelien
Dipl.-Inform. (FH) Jens Homann
(Hrsg.)

an der
Fakultät Informatik der Technischen Universität Dresden



Gefördert von der Klaus Tschira Stiftung,
gemeinnützige Gesellschaft mit beschränkter Haftung

sowie unter Mitwirkung der
GI-Regionalgruppe Dresden

am 28./29.10.1999
in Dresden

B.4. Stabilität und Sicherheit im Web – Der Test webbasierter Anwendungen

Dr. R. Schröder
Bode Management Consultants, Hamburg

Zusammenfassung

Dieser Artikel zeigt auf der Basis von Beispielen aus der Beratungspraxis die Auswirkungen webbasierter Systeme auf den Softwareentwicklungsprozeß im Allgemeinen und den Test im Besonderen. Er stellt praktische Lösungsmöglichkeiten für die neuen Anforderungen dar. Der Artikel stützt sich auf die praktischen Erfahrungen, die in der Beratungspraxis des Autors gesammelt wurden.
Dr. Roland Schröder ist einer der Geschäftsführer der Bode Management Consultants aus Hamburg. Das 1996 gegründete Unternehmen hat sich auf die Organisations- und Technologieberatung von Versicherungsunternehmen spezialisiert und erzielte 1998 bereits einen Umsatz von 2,3 Mio. DM.

1 Trends und ihre Auswirkungen im e-Commerce Bereich:

In der Versicherungsbranche, wie auch in vielen anderen Branchen wird die direkte Distribution über das Internet den klassischen Vertrieb ergänzen und teilweise ablösen. Die neuen Medien bieten neue Möglichkeiten der Integration der Fachabteilungen in die Kundenbeziehungen, der Content der Webanwendungen wird zunehmend in den Fachabteilungen generiert. Es kommt zu einer wesentlichen Beschleunigung der fachlichen Prozesse und zu einer „Amerikanisierung" der Zeit. Das heißt, die Mitarbeiter sind rund um die Uhr an den unterschiedlichsten Orten erreichbar.
In den Unternehmen kommt es im Rahmen dieser Entwicklungen zu einer zunehmenden virtuellen Integration und einer Verflachung der Wertschöpfungskette.
Ein bedeutsames Ziel der Versicherungsunternehmen in der Gegenwart ist die Steigerung der Cross Selling Rate in der Branche und Branchenübergreifend. Im Rahmen der Zielerreichung werden die Unternehmen zunehmend in Symbiose mit anderen Angeboten aktiv. So werden derzeit Kfz-Policen im Rahmen von Automärkten, Lebensversicherungen mit Konzertkarten, Gepäck-, Kranken- und Lebensversicherungen gemeinsam mit Urlaubsreisen angeboten. Das Ziel ist dem Kunden möglichst ein „Servicepaket" anzubieten.
Die Versicherungsunternehmen werden in den neuen Medien mit einem veränderten Kunden- und Nachfrageverhalten konfrontiert. Der derzeit zu beobachtende Trend zum

Markt- und Wettbewerbsorientiertem Verhalten der Kunden setzt sich im Web mit einer gesteigerten Dynamik fort.

Für die Unternehmen ist das Internet eine weitere Möglichkeit im Vertriebsmix, um die Kunden zu erreichen. Zielgruppen können durch personalisierte Webseiten gezielter angesprochen werden. Gleichzeitig dient das Medium zur Unterstützung der bisherigen Vertriebswege.

Die erwarteten Vorteile des e-business für die Versicherungen als Finanzdienstleister liegen in folgenden Bereichen:

- Marktbearbeitung ausweiten. Gewöhnlich treten die Kunden eher selten und zu unangenehmen Zeitpunkten mit ihrer Versicherung in Kontakt. Durch den Vertrieb von Gebrauchtwagen, Konzertkarten oder MP3-files über die Webseiten der Unternehmen läßt sich das ändern.
- Vertriebskosten senken. Den immensen Kosten für den Aufbau und den Betrieb von Webanwendungen stehen auf der anderen Seite geringere Kosten für Provision und Marketingaktionen gegenüber
- Transaktionskosten senken. Der Kunde als Sachbearbeiter ermöglicht erhebliche Einsparungen im Vertrieb und der internen Verarbeitung der Informationen. Dem stehen auf der anderen Seite jedoch steigende Kosten für Dublettenbereinigungen gegenüber.
- Segmentspezifische und kundenspezifische Vertriebssteuerung. Personalisierte Webseiten mit zugeschnittenen Angeboten für den einzelnen Kunden ermöglichen eine Erhöhung der Cros-Selling-Rate und Kundenbindung.
- Controllingfähigkeit der Produkte steigern. Der Vertrieb über das Web ermöglicht eine bessere und einfachere Zuordnung der anfallenden Kosten auf Kostenarten, Kostenstellen und Kostenträger.
- Bequemlichkeit für den Kunden. Die Leistungen und Informationen sind an 7 Tage die Woche über 24 Stunden ortsunabhängig verfügbar.
- Preis. Kostenvorteile können über den Preis der Produkte an die Kunden weitergegeben werden.
- Transparenz der Angebote. Auch in einem zunehmend differenzierten Markt kann über Internet-Tools (Webagents) und Makler die Transparenz für den Kunden hergestellt werden
- Aktualität. Die Implementierung neuer Produkte und Angebote ist im Internet in einem Bruchteil der Zeit möglich. Auch die Reaktion auf Kundenanfragen läßt sich wesentlich schneller realisieren.
- Servicequalität. Durch die direkte Ansprache im Zusammenspiel mit den oben genannten Faktoren, läßt sich ein neues Servicelevel realisieren.

- One-Stop-Shopping. Es besteht die Möglichkeit den Verkauf von der Information über das Angebot, den Antrag bis zur Policierung in einem Prozeß zu realisieren

Die im einzelnen erwarteten Ergebnisse zu den oben genannten Punkten sollten in einem Internet Value Audit dokumentiert werden und im Rahmen der Tests der Geschäftsprozesse validiert werden.

Der direkte Vertrieb über das Netz stellt somit die bisherigen Geschäftsprozesse und damit auch das bisherige Kerngeschäft in Frage.

Ein Teil der Produkte der Unternehmen ist sehr komplex und bedarf der Zusammenarbeit verschiedener Dienstleister mit den entsprechenden technischen Voraussetzungen (z. B. Gesundheitscheck für die Lebensversicherung).

Die Kreativität und Aktualität ist in den neuen Medien eine absolute Notwendigkeit und zugleich häufigste Fehlerquelle.

In der Tendenz geht die Entwicklung auch dahin, daß die Unternehmen untereinander leichter vergleichbar und damit austauschbarer werden.

In der Konsequenz wird der Druck zur Einführung von e-Commerce kurzfristig von den Mitbewerbern ausgehen und erst langfristig von den Kunden.

Nach meiner Einschätzung bildet der Vertrieb über das Internet den härtesten Vertriebsweg. Das ist dadurch bedingt, daß das Feedback durch den Kunden sehr gering bzw. normiert ist. Das Kundenverhalten läßt sich dadurch schwer beurteilen und die Möglichkeiten zur Einflußnahme sind gering. So entscheidet ein Kunde im Schnitt in den ersten 10 Sekunden, ob er auf einer Webseite bleibt oder weitergeht. Entscheidend sind solche Attribute wie Bequemlichkeit, Schnelligkeit und Zusatznutzen. Der Kunde als Sachbearbeiter zeichnet sich durch eine wesentlich geringere Loyalität gegenüber dem dienstleistenden Unternehmen aus. ‚Diese Erfahrung mußte zum Beispiel auch der Karstadt-Konzern machen. Das Webangebot „My World" erlebt gerade den dritten Relaunch, für den inklusive der Online Stops 46 Mio. Mark vorgesehen sind. Nach eigenen Angaben führte eine falsche Einschätzung des Kundenverhaltens zu dem wenig erfolgreichen Mall-Konzept mit vielen kleinen Geschäften. Das wird gegenwärtig zu einem ganzheitlichen Ansatz korrigiert. Auch die Beschränkung auf einen Ausschnitt aus dem Angebot (z. B. 100 Top CD's) führt zur Teilnahme am globalen Preiswettbewerb. Korrigiert wird das derzeit über ein Sortiment mit 200.000 CD's und dementsprechend höheren Preisen. Dieser Wechsel bedingte eine komplette Umstellung der softwaretechnischen und hardwaretechnischen Basis (Unixsystem vs. Host) sowie der engen Einbindung des Webgeschäfts in die Logistikkette des Unternehmens. Nach Einschätzung von „My World" ist im Webgeschäft die Geschwindigkeit am Bildschirm **und** in der nachfolgenden Ausführung Trumpf.' [COW 99].

Erschwerend für die Gestaltung von Webanwendungen wirkt sich der Faktor aus, daß derzeit ein relevanter Anteil der Nutzer hinter Proxy und Firewalls auf das Netz zugreifen und damit weitere Möglichkeiten der Einschränkung bzw. Verfälschung gegeben sind. Das betrifft insbesondere Java-Aplets, aber auch Zertifizierungen.
Auf der technischen Seite sind die Herausforderungen nicht minder anspruchsvoll. So haben wir es mit einer größeren Anzahl an unterschiedlichen Darstellungsmedien zu tun. Dazu gehören zum Beispiel die unterschiedlichen Internetexplorer, Navigatoren, Mosaik, LE 370-Browser, Lotus-Notes Clients, Handy, Palm und Set-Top-Boxen für Fernseher.
Weiterhin laufen Webanwendungen gewöhnlich 7 Tage die Woche und 24 Stunden am Tag. Damit entstehen massenhafte Anforderungen an das Operating, an die Verfügbarkeit und das Know-how wie sie in der Vergangenheit nur in wenigen meist Großrechenzentren üblich waren.

2 Das Vorgehen zur Entwicklung webbasierter Anwendungen

Die Entwicklung webbasierter Systeme kann sich weitgehend auf die erprobten, klassischen Verfahren der Softwareentwicklung und des Softwaretest stützen. So können je nach Entwicklungsweg und Ziel u.a. das V-Modell, das Spiralmodell, rapid Prototyping und auch die Vorgehensmethoden der objektorientierten Entwicklung zum Einsatz kommen.
Durch die gravierenden Auswirkungen auf die gesamte Wertschöpfungskette und die größerer Flexibilität und Verantwortlichkeit der Beteiligten, ist jedoch die Einbeziehung der Geschäftsprozesse in den Entwicklunsgprozeß noch wichtiger als bisher. So ist eine klare Definition der Ziele und deren permanente Überprüfung und Anpassung eine kritische Voraussetzung für den Erfolg im Netz. Durch den Einsatz von flexiblen Lösungen sinkt der Entwicklungsaufwand, erhöht sich die Flexibilität und steigt gleichzeitig die Notwendigkeit für die permanente Überprüfung der Systeme und für einen strukturierten Test. Während wir bei klassischen Hostprojekten noch von einem Anteil von ca. 20% Test am Gesamtaufwand ausgegangen sind, beträgt dieser Anteil bei gut gemanagten, oobasierten Client-Server-Projekten in der Regel 40 – 50 %. Wie dargestellt erhöht sich dieser Anteil bei Webprojekten durch

- die unmittelbaren Auswirkungen auf die Geschäftstätigkeit des Unternehmens(Kunde als Sachbearbeiter, Reaktionszeiten, Kosten von Fehlern etc.),
- die technische Vielfalt (Betriebssysteme, Browser, Firewalls etc.) und
- die Ansprüche an den Betrieb (7*24, Sicherheit, Verfügbarkeit, Performance etc.)

weiter. Die dazu kommenden neuen, bisher nicht bekannten Entwicklungswerkzeuge, Methoden und Verfahren können bei definierten (Mindest-)Projektumfang, festen

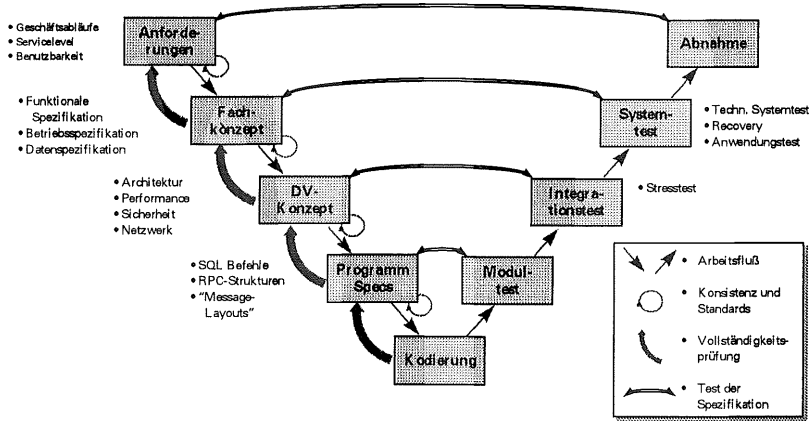
Endterminen, begrenzten Budgets und eingeschränkten Ressourcen zu den bekannten Effekten – Reduzierung der Tests und Sinkende Anwendungsqualität – kommen. Dieses, auch unter dem Begriff „Bananensoftware, reift beim Kunden" bekannte Vorgehen, kann aber gerade im Netz für ein Unternehmen schnell tödlich werden.

Ein Beispiel aus der Praxis: ‚Die Firma Dell, einer der größten Anbieter von PCs und Komponenten im Netz hat vor 3 Jahren einen Monitor zu 0 Mark auf seiner Webseite angeboten. Es wurden ca. 100 Monitore zu diesem Preis geordert. Telefonisch wurden die Besteller durch Mitarbeiter von Dell angerufen, und im Rahmen einer Entschuldigung über den Fehler und den wahren Preis informiert. Derzeit hat Dell Online mehr als 200.000 Zugriffe pro Woche und ein relevanter Teil des Umsatzes wird auf diesem Weg erzielt. Heute wären die Auswirkungen kaum noch zu managen.

Im Gegensatz zu klassischen Systemen, die in der Regel durch Mitarbeiter der eigenen Firma bedient werden, ist im Internet jeder Fehler für den Kunden sofort erkennbar und kann gravierende Auswirkungen haben.'[ECK 99].

Versicherungsunternehmen, die im wesentlichen nur Informationen produzieren, die sich auf einem Stück Papier, der Police materialisieren, stellen im Internet ihr Innerstes, die Qualität ihrer Informationsverarbeitung dar. ‚Nach wie vor geben die Unternehmen Millionen von Mark für neue (Internet-)Software aus und versagen daran, sie adäquat zu testen, wenn sie fertiggestellt ist. Die Software wird dann mit Fehlern in die Produktion übergeben. Untersuchungen des Quality Assurance Institute haben in den letzten Jahren ergeben, daß produktive Software im Schnitt Drei bis Sechs Defects auf tausend Lines of Sourcecode enthält.' [PER 95, Seite 29] Nichts deutet darauf hin, daß die Anzahl der Fehler in der modernen Internet-Software geringer geworden ist.

Am Beispiel des V-Modells sollen einige Charakteristika des Testvorgehens dargestellt werden:



Das Schema zeigt, daß im Verlauf eines Projektes sehr früh Testaktivitäten gestartet werden können. Mit der Entwicklung des Testplanes für den System- und Abnahmetest kann beispielsweise schon in der Phase "Fachliche Analyse" begonnen werden. Für jede Teststufe muß die dazugehörige Dokumentation bereitgestellt bzw. vorhandene Dokumentation erweitert werden.

Auf jeder Stufe können unterschiedliche Teststrategien Anwendung finden. Selbstverständlich ändern sich dabei auch die Testfälle. Die prinzipielle Vorgehensweise beim Testen und dazu erforderliche Dokumentation bleibt jedoch auf allen Teststufen gleich; es können jedoch für bestimmte Testarten auch besondere Maßnahmen nötig sein (z.B.: Ändern von Testtreibern).

Entsprechend der festgelegten Reihenfolge werden beim Modultest die Module und/oder ihr Zusammenspiel auf Lauffähigkeit getestet. Anwendbare Teststrategien sind Black- und White-Box-Test.

Beim Modultest ist es in der Regel erforderlich sich sogenannte „Testtreiber" (Webseite, die das zu testende Modul aufruft) und Stubs (Platzhalter für noch nicht fertiggestellte Module) zu entwickeln, um Tests auf dieser Ebene überhaupt durchführen zu können.

Weitere Ziele beim Modultest sind:

- Messen der Testabdeckung (White -Box Test)
- Messen des Ressourcenverbrauchs
- Messen der Qualität
- Sichtbarmachen der Daten

Subsystemtest / Integrationstest: Je nach Komplexität des Gesamtsystems, kann es erforderlich sein, daß das Zusammenspiel mehrerer zusammengehöriger Module in einem sogenannten Subsystem getestet wird. Die schrittweise Integration wird empfohlen, da die Fehleranalyse, auf der jeweils kleinen Anzahl neu integrierter Module, erleichtert wird. Hinsichtlich der Integrationsreihenfolge der verschiedenen Module kann eine "Top-Down" oder "Bottom-Up" Strategie verfolgt werden. Dabei liegt der Schwerpunkt beim Austesten der Schnittstellen zwischen den Modulen, aber auch der Test auf die Erfüllung der fachlichen Anforderungen erfolgt hier. Der Integrationstest wird durch den Programmierer / Fachbereich durchgeführt und dokumentiert.

Im Systemtest ist die Funktionalität des Gesamtsystems gegen die Anforderungsspezifikation abzugleichen. Der Systemtest beinhaltet auch den Test mit seltenen, oft nur theoretisch möglichen Grenzwerten, sowie den Test von Fehlerkonstellationen.

Den Abnahme- / Akzeptanztest sollte es über den Installationstest hinaus geben, insbesondere wenn die zuständige User im Vorfeld nicht oder nur unzureichend

eingebunden waren. In dieser Phase testet der Anwender (Endbenutzer) abschließend das System und prüft, ob seine Erwartungen gemäß den gestellten Anforderungen erfüllt worden sind. Wenn hier Erwartungen nicht erfüllt werden, sind bereits im frühen Stadium der Systemanforderungsdefinition Fehler gemacht worden. Um zu vermeiden, daß erst in der letzten Testphase gravierende Fehler in der Gesamtkonzeption festgestellt werden, sind Vertreter der Anwender in die Projektarbeit einzubeziehen.

Unter Testbeständen versteht man Daten, die zur Ausführung des Tests zusammen mit dem zu testenden Programm verwendet werden. Testbestände können Testfälle (siehe unten) beinhalten und sind damit entscheidend für den Verlauf des Tests.

Testbestände werden für die gestellten Forderungen immer einen Kompromiß darstellen:

- sie sollen repräsentativ sein, d.h. ein ausreichendes Abbild der Produktionsbestände darstellen
- sie sollen vollständig sein, d.h. sie sollen die Funktionsfähigkeit abdecken für Normalfälle, Sonderfälle, Ausnahmesituationen und Fehlersituationen
- sie sollen den Test sämtlicher Teilfunktionen ermöglichen
- sie sollen den Test sämtlicher Übergabebereiche von Schnittstellen zwischen Funktionen, Systemen, Systemteilen, Subsystemen ermöglichen
- sie sollen bei maschinellen Tests mit einem vernünftigen Ressourcenverbrauch auskommen
- sie sollen bei manuell zu prüfenden Tests eine nicht zu große Zahl von zu prüfenden Testfällen enthalten

Die Testbedingungen bilden die Grundlage für die zu erstellenden Testfälle. Sie umfassen, entsprechend der jeweiligen Teststufe, die durch die zu testende Einheit zu erbringende Funktionalität. So werden auf der Ebene des Modultest entsprechend den Designvorgaben alle Statements, und Datenzustände in die Testbedingungen aufgenommen. Auf der Ebene des Abnahmetest bilden die Geschäftsvorgänge und Ihre Ausprägungen die Grundlage für die Erstellung der Testbedingungen.

Der Entwurf von Testfällen für den Testbestand ist eine anspruchsvolle Aufgabe, da hiervon die Qualität des Tests und damit die des gesamten lauffähigen Systems abhängt. Unter einem Testfall versteht man folgendes Paar:

- die kompletten Eingabedaten
- alle dazugehörigen Ergebnisdaten

Die Eingabedaten müssen wohlüberlegt sein. Mit jedem Testfall versucht man einen bestimmten Fehlertypus bei der Testdurchführung sichtbar werden zu lassen. Zufallsgeneratoren helfen aus diesem Grund wenig bei der Formulierung von

Testfällen. So erzeugte Testfälle verschwenden Rechenzeit, ohne die Ergebnisse aussagekräftiger zu machen.
Die Funktion eines Moduls ist nur durch das Paar Eingabe / Ausgabe zu beschreiben. Daher sind Testfälle wertlos, deren erwartete Ergebnisse nicht vor der Testdurchführung spezifiziert sind, denn sie enthalten keine Aussage über die Korrektheit der Ergebnisse.
Testzyklen sind die Zusammenfassung von Testfällen nach gleichen Merkmalen. Sie dienen zur Strukturierung und Koordinierung der Testerstellung und Testdurchführung.

Softwarequalität und Qualitätsmerkmale: Qualität ist keine Eigenschaft, die ein Softwareprodukt hat oder nicht hat, sondern es ist festzulegen, welche Qualitätsanforderungen an ein Produkt gestellt werden, wobei zu prüfen ist, ob gestellte Anforderungen in der zu Verfügung stehenden Zeit und den zur Verfügung stehenden Mitteln zu realisieren sind. Alle Testaktivitäten gehören in den großen Rahmen Qualitätssicherung einer Software mit all ihren Facetten. Dazu gehören meines Erachtens neben anderen die Aspekte: Design for Quality, Design for Operations, Datenmanagement, Kommunikation, System, Performance, Anwendung und Test. Vgl.: [GLA 92]. Allerdings würde eine umfassende Betrachtung der Qualitätskriterien (von denen nur eine Facette der Test ist) leider den Umfang des Vortrages sprengen. Die Qualität eines Softwareproduktes läßt sich nicht als ein einzelner Wert ermitteln. Vielmehr muß eine Differenzierung hinsichtlich verschiedener Kenngrößen stattfinden. Für jede dieser Kenngrößen können Qualitätsanforderungen definiert werden. Man unterscheidet folgende Qualitätsmerkmale:

Vgl.: [CAR 99, Seite 8 ff]
Die ersten sechs Qualitätsmerkmale sind bei der Durchführung von Tests für Websysteme von besonderer Bedeutung.
Qualitätsmaße dienen dazu Qualitätsmerkmale in konstruktive Maßnahmen bei der Softwareentwicklung und in analytische Maßnahmen hinsichtlich der Erfüllung von

Qualitätsanforderungen umzusetzen. Das bedeutet, daß Qualitätsmaße die angewandten Qualitätsmerkmale meßbar machen.
Man unterscheidet hierbei zum Beispiel in:

- Entwicklungsprozeßbezogene Qualitätsmaße
  - Testabdeckungsmaße
  - Maße zur Erfüllung von Checklisten
- Software-Produktbezogene Qualitätsmaße
  - Komplexität nach McCabe
  - Programmlänge nach Halstead
  - strukturelle Komplexität
  - hierarchische Komplexität
- Betriebsbezogene Qualitätsmaße
  - Verhalten der Software im realen Betrieb (z.B. Anzahl Fehler pro Zeiteinheit)

Qualitätsziele dienen für Maßnahmen der Softwareentwicklung (z.B. Testziele für Testaktivitäten), sowie als Freigabe- und Abnahmekriterium im Rahmen der Kontrolle. Außerdem stellen Qualitätsziele die konkreten Qualitätsanforderungen dar. In der Testdokumentation (Testkonzept, Testplan) sind sinnvoll erscheinende Werte für die Qualitätsmaße zu bestimmen.
Qualitätsanforderungen ergeben sich aus den Anforderungen an das Softwareprodukt selbst.

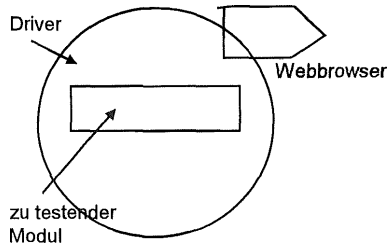
Unter Teststrategien ist in diesem Zusammenhang die planvolle Vorgehensweise des Testens zu verstehen. Die Reihenfolge der Testaktivitäten wird durch die gewählte Methode des Systementwicklungsprozesses beeinflußt. Die Strategie der Systemintegration und die Durchführung des Integrationstests wird bereits in frühen Phasen des Systementwurfs entwickelt. [Vgl. PER 95, Seite 178]
Das Ziel der Teststrategie ist, zu beweisen, daß die Anzahl der Fehler bzw. Abweichungen eines realisierten Systems oberhalb einer definierten Schwelle liegt.
Bei Einsatz der Ganzheitsstrategie wird ein bereits fertiggestelltes Programm als Gesamtheit ausgetestet. Die Einsatzmöglichkeit liegt beim Test von Programmen, die nicht zu umfangreich bzw. sehr klar strukturiert sind.
Das Testen nach der "Bottom-Up"-Methode bedeutet, daß mit den Modulen auf unterster Ebene begonnen wird und erst danach die Steuerung erstellt und ausgetestet wird. Für diese Teile ist die Umwelt (z.B. Eingabeschnittstellen und Schnittstellen zu anderen Systemen) zu simulieren. Eine Integration zum Gesamtsystem ist in der Regel erst nach Fertigstellung aller Funktionen möglich.

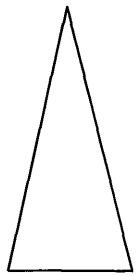
Der Einsatz sollte primär bei sehr komplexen Programmen erfolgen, deren Steuerung der Module immer im hierarchisch darüberliegenden Modul liegt. Für den Test werden Driver (oder Testbett) benötigt. Der Driver sorgt für die Möglichkeit der Eingabe von Daten über die definierte Schnittstelle an das aufgerufene Modul. Diese Eingabe sollte nach Möglichkeit Online erfolgen. Nach der Verarbeitung dieser Daten im Modul übernimmt der Driver die zur Verfügung gestellten Ausgabedaten (Ausgabeparameter). Nach Abschluß der Test sind sowohl die Eingabe- als auch die Ausgabeparameter auszugeben.

In einer Testbibliothek sollten die, für die Eingabe bereitzustellenden Testdaten abgelegt werden.

Bei der „Bottom-Up"-Methode können u.U. Probleme mit der Versorgung der Schnittstellen auftreten. Die Vorteile dieser Vorgehensweise liegen

- in unabhängigen Entwicklung und Test von Programmteilen
- in leicht verständlichen Testresultaten und
- in weniger Testläufen.





Vgl.: [ZIM 89], [MAR 95]

Bei der "Top-Down"-Methode hingegen wird der Test von oben (z.B. mit der Web-Steuerung) begonnen und dann durch schrittweise Aufnahme weiterer Module das System nach und nach komplettiert. Noch nicht integrierte Module müssen durch mehr oder minder intelligente Hilfsroutinen (DUMMY-Module) simuliert werden. Es bietet sich an, bei den Modulen die Funktion entsprechend ihrer Wichtigkeit zu testen, damit evtl. auftretende Terminprobleme leichter gelöst werden können.

Der Einsatz bietet sich an für den Test von Programmen, die sehr klar strukturiert sind und deren Modulsteuerung immer im hierarchisch darüberliegenden Modul liegt..

Als Hilfsmittel werden DUMMY-Module und Testbibliotheken benötigt.

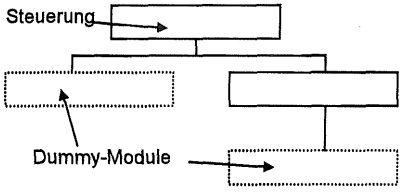
Ein Dummy-Modul kann sowohl Daten über eine Schnittstelle erhalten (Eingabeparameter) als auch Daten über eine definierte Schnittstelle weitergeben (Ausgabeparameter).

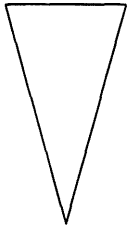
Jedes Dummy-Modul soll über die Möglichkeit der Ausgabe der Modulbezeichnung verfügen und muß instrumentiert - Definition eines Durchlaufzählers - werden.
Die Vorteile sind:

- der Integrationstest wird immer wieder ausgeführt
- Fehler werden vom neu hinzukommenden Code verursacht
- Detaillieren, Codieren und Testen werden überlappt durchgeführt

Nachteile:

- Tiefer gelegene Module sind schwieriger zu erreichen
- Testresultate sind oft schwer verständlich





Vgl.: [ZIM 89], [MAR 95]
Bei der Code-Inspektion handelt es sich um eine Strategie, die auf dem Besprechen des Programmcodes innerhalb eines Gremiums basiert. Der Entwickler erklärt dabei den anderen Mitgliedern des Gremiums / Teams die Funktionsweise seines Moduls oder Programmes. Die Mitglieder des Gremiums stellen dazu kritische Fragen und versuchen dabei Schwachstellen oder mögliche Fehlerquellen zu finden. Jeder gefundene Mangel wird protokolliert und im Anschluß daran an den Entwickler ausgehändigt. Die Zielsetzung ist zum einen die Qualität der Software zu verbessern und zum Anderen die Überprüfung des Codes auf Übereinstimmung mit dem Feinentwurf. Mögliche Prüfbereiche bei einer Code-Inspektion sind:

- Schnittstellen des Prüfobjektes
- Ablaufstruktur des Programmes
- die Verwendung von Variablen bzw. deren Namen
- Berechnungsformeln
- Ein-/ Ausgabe
- Kommentare
- Einhaltung von Codierstandards

Strategien für den Entwurf der Testfälle: Beim Black-Box-Test wird das Testobjekt als Funktionseinheit betrachtet, die auf bestimmte Eingaben mit spezifischen Ausgaben reagiert. In diesem Fall sind neben den für den Anwender wichtigen Standardfällen auch auf Grenzfälle zu achten. [KAN 88, Seite 36 ff] Der Black-Box-Test leitet sich aus der Spezifikation ab und die ausgewählten Testfälle sind somit von der Art der Implementierung unabhängig.

Die wichtigsten Black-Box-Methoden sind:

- In der Methode der Funktionsabdeckung werden anhand konkreter Anwendungen die Funktionen des Testobjekts identifiziert. Dazu wird für jede Funktion eine Ein- / Ausgabe-Spezifikation erstellt. Mit den auf dieser Spezifikation beruhenden Testfällen werden Tests durchgeführt, um zu zeigen, daß die Funktionen vorhanden und auch ausführbar sind. Ein sehr sinnvolles und nützliches Hilfsmittel für die Zusammenfassung von Testfällen, bei der Anwendung der Funktionsabdeckung ist eine Testfallmatrix.
- In der Methode der Äquivalenzklassen repräsentiert eine Äquivalenzklasse eine Menge von Werten einer bestimmten Größe. Beim Test wird dann davon ausgegangen, daß ein Wert aus dieser Menge (Klasse) stellvertretend für einen beliebigen anderen Wert dieser Klasse verwendet werden kann, um eine Fehlerart aufzudecken.

  Die Wertebereiche der Ein- und Ausgabegrößen werden in Äquivalenzklassen eingeteilt. Dabei sind zu jeder gültigen Äquivalenzklasse auch ungültige zu wählen. Aus jeder Äquivalenzklasse wird ein Wert ausgewählt (Bestimmung des Testfalls) Die Zerlegung der Äquivalenzklassen muß auf ihre Vollständigkeit geprüft werden.
- Die Grenzwertanalyse ergänzt die Äquivalenzklassenmethode, indem sie die Grenzen der Wertebereiche von Ein-/ Ausgabegrößen oder ihrer Umgebung abdeckt. Weitere Informationen in [MAR 95, Seite 145ff]

Die White-Box Testmethoden beruhen auf Kenngrößen, die einerseits die Testabdeckung und andererseits die Strukturkomplexität des Testobjekts betreffen. Sie ist im Gegensatz zum Black-Box-Test (Funktionaler Test) eine strukturelle Testtechnik. Strukturelle Testtechniken vergleichen das Verhalten der Programme gegen die im Source-Code abgelegten Absichten.

Damit scheint der White-Box-Test unsicher gegenüber dem Black-Box-Test, da er Fehler in der fachlichen Umsetzung nicht finden kann. Aber fachliche Beschreibungen existieren häufig nicht oder nicht vollständig. Das trifft insbesondere auf das Ende eines Entwicklungszyklusses zu, wenn die Anforderungsspezifikationen seltener aktualisiert werden und das aktuelle Produkt selber die Rolle der Spezifikation übernimmt.

Die Analyse der Testabdeckung ist der Prozeß:

- Des Finden's von Programmteilen, die durch keine Test abgedeckt sind,
- Der Erzeugung von zusätzlichen Testfällen zur Erhöhung der Testabdeckung und
- Der Festlegung von quantitativen Kennziffern für die Testabdeckung, die eine indirekte Kennziffer für die Qualität sind.

Ein optionaler Aspekt der Analyse der Testabdeckung ist;

- Das Finden von redundanten Testfällen, die die Testabdeckung nicht erhöhen.

Bei den Testmethoden, die auf Abdeckungskenngrößen basieren, wird häufig die Ablaufstruktur des Testobjekts (Programm / Modul) als Graph dargestellt.

Die als Knoten dargestellten Anweisungen und der durch Linien gekennzeichnete Kontrollfluß ergeben einen Programmgraphen, der praktisch alle möglichen Ablaufpfade enthält. Da es in der Praxis sehr aufwendig ist, alle Programmpfade zu durchlaufen, begnügt man sich mit dem Erreichen von Zielwerten für die verschiedenen Testabdeckungskenngrößen.

Folgende Testabdeckungskenngrößen sind gebräuchlich:

C_0 : ***Anweisungsabdeckung***

Darunter versteht man das Verhältnis der Anzahl an durchlaufenen Anweisungen zur Gesamtanzahl der Anweisungen eines Testobjekts.

C_1 : ***Zweigabdeckung***

Darunter versteht man das Verhältnis von durchlaufenen Zweigen zu allen möglichen Zweigen des Testobjektes.

C_2 : ***Bedingungsabdeckung***

Wie C_1, aber statt Zweigen werden Terme innerhalb von Ausdrücken verwendet. Eine Testabdeckung von 100% C_2 bedeutet, daß in einem Programmabschnitt innerhalb jedes Ausdrucks einer Bedingungsanweisung oder Schleifenanweisung jeder Term mindestens einmal evaluiert wurde.

Diese Abdeckung entspricht gut den Anforderungen von C, C++ und Java.

C_3 : ***Abdeckung aller Bedingungskombinationen***

Um eine Testabdeckung von 100% C_3 zu erhalten, müssen alle möglichen Kombinationen von Elementarbedingungen innerhalb einer Abfrage oder Schleifenbedingung einmal durchlaufen werden.

C_4 : ***Pfadabdeckung***

Alle möglichen Pfade eines Moduls werden zumindest einmal durchlaufen.

Weitergehende Beschreibungen zu den einzelnen Kennziffern finden sich in [BEI 90 p75ff] und [ROP 94 p 39ff].

Die Zielwerte der Testabdeckungskenngrößen müssen für ein Projekt festgelegt werden. Typische Beispiele aus der Praxis sind eine Anweisungsabdeckung (C_0) von 95% und eine Zweigabdeckung (C_1) von 85%.

Eine Testabdeckung von 100% ist jedoch noch keine Garantie dafür, daß das Testobjekt fehlerfrei ist. Insbesondere fehlende Pfade, aber auch Fehler in der Spezifikation, können durch diese Art von Testen nicht entdeckt werden.
Die Auswahl von geeigneten Testkennziffern kann die Produktivität des Testens in einem erheblichen Maß erhöhen. Die höchste Testproduktivität besteht darin, möglichst viele Fehler mit möglichst wenig Testfällen zu finden.
Eine Strategie, die zu schnellen Erfolgen führt ist der Einsatz der Kennziffern für komplette Programme und erst im Anschluß für einzelne Module und Komponenten.
Der White-Box-Test bzw. die Ermittlung der Testabdeckungskenngrößen wird in der Praxis durch geeignete Testtools unterstützt. Auf dem Markt existiert eine Reihe von Werkzeugen zur automatisierten Berechnung der Testabdeckung. Diese Werkzeuge dienen zur Sicherung der Qualität der Testfälle, nicht zur Qualitätssicherung des aktuellen Softwareproduktes. Nicht für jedes neue Softwarerelease wird der Einsatz eines "Coverage-Analyseres" benötigt.
Der Einsatz dieser Werkzeuge benötigt den Zugriff auf den Source-Code und zusätzliche Rechenzeit.
Der White-Box Test als eine von vielen Testtechniken sollte insbesondere bei Webprojekten nicht alleine angewendet werden.
Der Belastungstest oder auch Stress-Test, will das Verhalten des zu testenden Programmes unter Last aufzeigen. Vor allem bei Websystemen, die Multi-User-Systeme sind, ist nachzuweisen, daß das System auch bei konkurrierendem Zugriff auf Ressourcen ordnungsgemäß arbeitet. Hierbei empfiehlt sich die Verwendung eines Werkzeugs, um im Falle eines Abbruchs, Anhaltspunkte für die Gründe des Fehlverhaltens zu erhalten.
Ein Testverfahren für den Test mit Testtools:
Referenztest: Erstellen von SOLL-Resultaten durch Programmausführung
Regressionstest (Re-Test): Wiederholung der Referenzläufe nach Modifikationen des Testobjekts (Wartung)
Vergleich: Vergleichen der Resultate aus Referenztest und Re-Test
Übernahme: Übernehmen der Re-Test-Resultate als Referenz für weitere Re-Tests
Fortsetzen: Nachträgliche Bearbeitung von Referenz-Resultaten.

3 Der Test von e-Commerce-Systemen

Der Test webbasierter System ist durch eine Reihe von Faktoren gekennzeichnet, die in dieser Konzentration bisher nur selten anzutreffen war. Dazu gehören

- die Auswirkungen auf die einzelnen Geschäftsprozesse,
- die enge Verknüpfung mit einer Vielzahl von zuliefernden und zu bedienenden Systemen,
- der Einsatz von DataWarehouse-Lösungen als Grundlage für die Personalisierung,
- die Nutzung unterschiedlichste Lösungen für die Frontends der User (Browser),
- die Sicherheitsanforderungen
- die Verfügbarkeit und Skalierbarkeit
- die Wiederholbarkeit der Tests bei erhöhter Flexibilität der Lösungen
- der Betrieb an 7 Tage und 24 Stunden.

Das bedingt die Verwendung von strukturierten, erprobten Verfahren für die Planung und Organisation des Test und den Einsatz von Tools und Werkzeugen für die weitgehende Automatisierung der Testdurchführung. Gegenwärtig existieren mehrere hundert Testtools, die spezielle Testaufgaben abdecken. Dabei ist der Korrektur- und Nachtestaufwand insoweit vom Testverfahren und dem gewählten Tool abhängig, als vorhandene Fehler früher, später oder erst im produktiven Einsatz gefunden werden. Dabei ersetzt kein Tool die Planung und Organisation, kann aber wie ein gut ausgewählter Hammer das Testvorgehen wesentlich beschleunigen und vereinfachen. Andererseits kann es aber auch den Test selbst unmöglich machen.' [PER 95, Seite 361]

Der Testprozeß unter Einsatz von Testtools gliedert sich in folgende Schritte:

- Planung der Tests und der notwendigen Ressourcen, Definition der Ziele, Erstellung eines Testmodells
- Entwurf von Testbedingungen, Testfällen und Testergebnissen
- Definition und Erzeugung von Testdaten, Erstellen eines Daten-Repositories
- Aufzeichnen und Anpassen von Testscripts (inklusive Checkpoints)
- Automatisierte Testausführung
- Analyse der Ergebnisse
- Fehlererkennung, -priorisierung sowie Fehlerkorrektur und Verfolgung
- Lasttest
- Planung und Unterstützung der Produktion

Das Ziel dieses Vorgehens ist die Schaffung eines wiederholbaren, automatisierten Prozesses durch den zielgerichteten Einsatz von Tools zur Reduzierung des Testaufwandes bei gleichzeitiger Erhöhung der Testabdeckung.

Die Erstellung eines Testmodells basiert auf den im Rahmen der Entwicklung erstellten Dokumenten und überprüft in diesem Zusammenhang die Umsetzung der Anforderungen. Dazu gehören:

Prozessmodell	Veränderungen in den Wertschöpfungsketten der Unternehmen, Umsetzung der qualitativen Kennziffern, bzw. bei einem iterativen Entwicklungsprozeß die Realisierung der Teilziele
Tätigkeiten	in den einzelnen Prozessen, durch Kunden und Mitarbeiter, die Einrichtung der Tätigkeiten, Hilfen und ggf. Handbücher
Implementierung	mit dem Test der Funktionalität der Webseiten etc.
Objekte	Java-Applets, Basisklassen, XML-Daten
Benutzbarkeit	Bequemlichkeit, Komfort, Antwortzeiten, Verfügbarkeit
Objektanalyse	Virtuelle Maschinen, Webbrowser-Implementierungen, Grafikfähigkeiten (Auflösungen, Größen, Farben)
Gebrauchsmodell	Sparten, Produkte, Abteilungen etc.

‚Die Zeit, die in die Testplanung investiert wird, wird normalerweise durch einen effizienteren Test um ein vielfaches ausgeglichen. Als Richtlinie gilt, daß ungefähr 30% der gesamten Testzeit für strategische und taktische Testplanung genutzt werden sollte.' [PER 95, Seite 41]

Die Testdefinition sollte in den Web-Projekten logisch vom Einfachen zum Komplizierten erfolgen. Durch den Einsatz von Standarsoftwarekomponenten kann der Test von Atomen, Collaborationen und Subkomponenten gekürzt werden. Nach unseren Erfahrungen sollte jedoch nicht vollständig auf einen Test dieser Standarsoftwarekomponenten verzichtet werden und der Test insbesondere bei Releasewechseln wiederholt werden.

Um komplexe Anwendungen in einem überschaubaren Zeitraum zu testen, wird technische Unterstützung benötigt. Es ist deshalb unerläßlich, insbesondere für Web-Projekte geeignete Werkzeuge zur Verfügung zu stellen.

Diese Werkzeuge sollen Tests im allgemeinen

- vereinfachen
- beschleunigen
- sicher machen
- dokumentieren
- transparent und nachvollziehbar gestalten
- bei der Generierung von Testdaten unterstützen
- Testsituationen simulieren.

Die speziellen Anforderungen an die einzelnen Werkzeuge ergeben sich aus der jeweiligen Testphase bzw. aus der Art der gerade durchzuführenden Arbeiten.

Durch eine Reihe von Softwareanbietern werden Tools und Verfahren unter anderem für

- die Testadministration sowie das Management des Testprozesses
- die Erstellung von Testbedingungen und Testfällen,
- die Aufzeichnung und das Abspielen von Testfällen,
- die Überprüfung von Input und Output (Checkpoints),
- die Überprüfung von GUI-Objekten und Styleguides,
- die Testwiederholung,
- das Fehlermanagement, (hier besteht ein enger Zusammenhang mit dem Anforderungsmanagement),
- die Messung der Testabdeckung,
- den Last- und Performancetest,
- die Überprüfung der Serververfügbarkeit und Systemressourcen sowie
- die Erstellung, Validierung und Verwaltung von Testdaten.

angeboten. Die Auswahl und Implementierung ist abhängig vom vorhandenen Umfeld und den gewünschten Zielen. Vgl.: [ROY 92] Seite 160 ff.

Das bedingt der Aufbau einer Testhierachie im Sinne der Testfälle (vom Einfachen zum Komplizierten) sowie Hard- und Software. Es werden Verfahren für das Testsetup und Cleanup benötigt. Die Testpläne bilden dann die Grundlage für die Erzeugung von Testsuites als Einheit von Testbedingungen, Testfällen und Ergebnissen). Die Grundlage für den Test bilden die Testdaten

Testmanagementwerkzeuge

Die für den Qualitätssicherungsprozess von Webanwendungen nützlichen Werkzeuge beinhalten im Allgemeinen Unterstützung für die folgenden Prozesse:

- Testfallplanung
- Testfallerstellung
- Testfallausführung
- Fehlerhandling und
- Fehleranalyse

Die Protokolle des Test-Managements geben Auskunft über den Teststatus, die Testaktivitäten und unterstützen bei der Bestimmung des Zeitpunktes für den produktiven Einsatz.

Im Rahmen des Error-Managements einer Web-Applikation werden die Abweichungen dokumentiert, klassifiziert und zugeordnet [Weitere Ausführungen in KAN 88, Seite 55 ff].Daraus können (und sollten) dann neben den "reinen" Testaktivitäten auch Auswertungen zur Softwarequalität durchgeführt und entsprechende Maßnahmen

umgesetzt werden. Die Systematik zur Klassifizierung der Error's (Fehler) ist nach meiner Erfahrung, neben den durch die Softwarearchitektur gegebenen Kriterien, abhängig vom jeweiligen Projekt, den Aufgaben, der Teamstruktur, der Größe des Korrekturteams, dem Zeitaspekt, der Anzahl der Fehler und weiteren "weichen" Faktoren. Dementsprechend ist im Vorfeld im Rahmen der Testplanung eine Strukturierung / Ablagesystematik der zu erwartenden Fehler notwendig. Gleichzeitig müssen in diesem Zusammenhang die Aufwände für die Korrektur der Fehler abgeschätzt und Regeln für die Abnahme der Software festgelegt werden.
Quasi-Standards für die Fehlerklassifikation können von Tivoli (IBM), von Microsoft, der OSI und auch der LINUX-Entwicklung genutzt werden.
Messung der Testabdeckung durch Test Coverage Monitor Tools zur Messung der Testeffektivität. Ein Programm gilt gewöhnlich als getestet, wenn ein Set von Eingabedaten die erwarteten Ergebnisse produziert. Dabei kann in der Praxis ein Test dazu führen, daß mit den Testfällen nur ein kleiner Teil des Codes wirklich ausgeführt wird. Diese Tools messen die Testabdeckung im Rahmen der C0-Abdeckung, C1-Abdeckung oder C2-Abdeckung. Gewöhnlich wird dazu der Code instrumentiert und während der Laufzeit ausgewertet. Ein Nachteil des dargestellten Verfahrens ist die Verfälschung der Programmlaufzeiten. Eine gleichzeitige Performancemessung in den Testumgebungen wird damit unmöglich bzw. sehr erschwert.
Hilfe in der vollständigen Erstellung von Testfällen bieten Tools zur Messung der Testabdeckung. Die Produktion von Testfällen ist häufig ein sehr willkürlicher oder sehr beschwerlicher Prozeß. Zudem gibt es insbesondere bei Black-Box-Test's häufig kein Feedback über die Qualität der Testfälle. Tool zur Messung der Testabdeckung kann bei dieser Aufgabe helfen indem er den getesteten und ungetesteten Code dokumentiert.

Test der Funktionalität / GUI-Test

Über den werkzeuggestützten Test der grafischen Benutzeroberfläche (GUI) werden sämtliche Funktionen und Menüs einer Applikation durchlaufen, um die Übereinstimmung mit den Anforderungen zu überprüfen. Eine, in den Werkzeugen, häufig genutzte Art diesen Test durchzuführen ist das Aufzeichnen aller Tätigkeiten, wie beispielsweise Tastatureingaben, Menü-Anwahlen und Maus-Aktionen. Die Aufzeichnungen werden in einem Skript hinterlegt, welches sich anschließend beliebig oft ausführen läßt. Die Tools bieten im allgemeinen die Möglichkeit der Parametrisierung und Verarbeitung von Variablen.

Last- und Performancetest

Die entsprechenden Tools setzen die Anwendung auf einem Applikations- und/oder Datenbank-Server unter Last. Es werden die Tätigkeiten auf dem Gesamtsystem

durchgeführt, die später im Produktiveinsatz von der Applikation erwartet werden. Dazu gehört, daß die Anzahl der Testbenutzer der realen Anzahl der Produktivanwender zumindest entspricht. Zunehmend werden diese Test-Tools mit Werkzeugen kombiniert die in regelmäßigen Abständen während des Betriebes einer Webanwendung die Verfügbarkeit und Performance messen. Häufig kommen dabei dieselben Werkzeuge zum Einsatz.
Eine Quantifizierung der Kosten für die Entfernung von Fehlern aus Webanwendungen ist mit meinem bisherigen Erfahrungsstand nicht möglich. Eine Erfahrungslage des Quality Assurance Institute ist in [PER 95] auf Seite 56 dargestellt.

4 Zusammenfassung

Bedingt durch die gravierenden Auswirkungen von personalisierten Internetdienstleistungen ist ein methodisches Vorgehen ein kritischer Faktor für dauerhaften Erfolg im Web. Der direkte Zugriff der Kunden und die Flexibilität der Software bedingen den Einsatz neuer, effizienter Lösungen für die Implementierung und den Test webbasierter Anwendungen.

Literatur

[BEI 90] Boris Beizer, Software Testing Techniques, 2nd edition, New York: Van Nostrand Reinhold, 1990

[BEI 95] Boris Beizer, Black-Box Testing – Techniques for Functional Testing of Software and Systems, John Wiley & Sons, Inc. , 1995

[CAR 90] David N Card with Robert L. Glass, Measuring Software Design Quality, Prentice Hall, 1990

[COW 99] Computerwoche Extra „Internet", 1999

[ECK 99] Scott Eckert, Direktor von Dell Online im forum ecommerce, ct 7/99, bzw. Bill Gates in Digital Business

[GLA 92] Robert Glass, Building Quality Software, Prentice Hall 1992

[JOR 95] Paul Jorgensen, Software Testing: a craftman's approach, CRC Press Inc, 1995

[KAN 88] Cem Kaner, Testing Computer Software, TAB Books Inc. 1988

[KAN 93] Cem Kaner, Jack Falk, Hung Quoc Nguyen, Testing Computer Software, Second Edition, Van Nostrand Reinhold, 1993

[MAR 95] Brian Marrick, The Craft of Software Testing, Subsystem Testing, Including Object-Based and Object-Oriented Testing, Prentice Hall, 1995

[OSK 97] Östen Oskarsson / Robert Glass, ISO 9000 und Softwarequalität, Prentice Hall, München, 1997

[PER 95] William E. Perry, Effective Methods for software testing, John Wiley & Sons, Inc. , 1995

[ROP 94] Marc Roper, Software Testing, London, McGraw-Hill Book Company, 1994

[ROY 93] Thomas C. Royer, Software Testing Management, Prentice Hall, 1993

[ZIM 89] Peter Zimmermann, Testtechniken, HMT 1989